AFIT/GAE/ENY/96D-3

MODEL PREDICTIVE CONTROL OF AEROSPACE SYSTEMS

THESIS

Derek W. Ebdon, Captain, USAF

AFIT/GAE/ENY/96D-3

AFIT/GAE/ENY/96D-3

MODEL PREDICTIVE CONTROL OF AEROSPACE SYSTEMS

THESIS

Presented to the Faculty of the Graduate School of Engineering

Air Education and Training Command

In Partial Fulfillment of the Requirements

for the Degree of

Master of Science in Aeronautical Engineering

Derek W. Ebdon, B.S.

Captain, USAF

December 1996

Acknowledgments

I would like to express my sincerest gratitude to my thesis advisor, Captain Sharon A. Heise. Her guidance and support were invaluable to me throughout this project. I would also like to express my appreciation to Dr. Brad Liebst, Dr. Brett Ridgley, and Lt Col Stuart Kramer for imparting to me some fraction of their knowledge and experience in the field of control systems and aircraft dynamics. Additionally, I would like to thank the Wright Laboratory's Flight Control Division of the Flight Dynamics Directorate for sponsoring in this thesis.

I cannot possibly thank my parents enough for all the help they've given me over the years, but to my father, David, and my mother, Priscilla, I offer my most heartfelt thanks. They gave me the chance to succeed. Lastly, I would like to thank my conjointe (Fr.), Hélène Paris, for her kindness and understanding throughout the long, torturous process of preparing this document. Her moral and nutritional support were key factors in the completion of my masters degree at AFIT.

Table of Contents

List of Figures

List of Tables

Notation

n	Order of the plant
ξ	Number of plant inputs
η	Number of plant outputs
κ	Order of the state estimator
p	Prediction horizon
q	Control horizon
r	Optimization horizon
ρ	$\max(p,q)$
E	Expectation operator
z^{-1}	Backward shift operator
Δ	Differencing operator, $\Delta = 1 - z^{-1}$
A^{T}	Transpose of the matrix A
A^{-T}	Transpose of the matrix A^{-1}
$\|x\|_2$	Euclidean 2-norm
$\|x\|_R^2$	Weighted square 2 norm, $x^{T}Rx$
$\mathbb{R}$	Set of real numbers
$\mathcal{RH}_\infty$	Set of all proper and real rational stable transfer matrices
$\mathcal{RL}_\infty$	Set of all proper and real rational transfer matrices with no poles on the unit circle
TVS	Symmetric Thrust Vectoring
AS	Symmetric Aileron
SS	Symmetric Stabilator
LES	Symmetric Leading Edge Flap
TES	Symmetric Trailing Edge Flap
T	Throttle

Abstract

Model Predictive Control (MPC) is the class of control methods that optimizes a specified performance index over a set of future inputs to minimize future output deviations from a specified trajectory, subject to system constraints. MPC operates on a receding horizon, calculating a series of future control inputs at each time step. The controller then implements the first input, discards the rest, and calculates the next series of inputs at the next time step. Because this control method involves on-line optimization, it has traditionally been applied mainly to low-bandwidth processes.

This research effort applies an MPC strategy to a high-performance aerospace system with the goal of exploiting the constraint handling qualities of MPC for fault-tolerant control. To demonstrate the effectiveness of MPC at handling control surface failures, one or more control surfaces on a high-performance fighter aircraft, primarily the F-18 High Alpha Research Vehicle, are rendered inoperable during longitudinal maneuvers. In general, simulated failures of a single control element occur at peak deflection, whereas the time and magnitude of multiple control element failures are selected on a case-by-case basis. Subsequent to the simulated failure or failures, the controller is forced to compensate by using the remaining control surfaces both to maintain stability and to attempt to retain near-nominal performance.

To accomplish these objectives, Matlab and proprietary predictive control Matlab routines are used to develop the functions and matrices necessary to implement an MPC controller. Simulations of aircraft performance subsequent to control surface failures are then accomplished using Simulink in order to determine controller effectiveness.

MODEL PREDICTIVE CONTROL OF AEROSPACE SYSTEMS

1.0 Introduction

1.1 Model Predictive Control

Model Predictive Control (MPC) is the class of control methods that optimizes a specified performance index over a set of future input moves in order to minimize the weighted future output deviations from a setpoint trajectory along a prediction horizon and the weighted control increment inputs implemented along a control horizon. At each discrete time step, this optimization process determines a series of future control input moves, implements the first move, and discards the rest. In the case of Stable Generalized Predictive Control (SGPC) or other formulations employing a stabilizing inner feedback loop, the output of the optimization is a series of "quasi-reference" signals instead of control increment inputs because the optimizer no longer feeds directly into the plant [2,3]. This introduction of the "quasi-reference" signal as the input to the inner loop also allows for an independent optimization horizon, r, because an optimal series of plant inputs are no longer being directly calculated. In addition, the inner feedback loop enforces an implicit terminal constraint, allowing stability to be achieved through the existence of a monotonically decreasing cost function.

MPC controllers have traditionally been used in slow industrial processes because they involve the use of on-line optimization. However, as computers become more

powerful, MPC will consequently become more practical in high bandwidth applications such as aerospace systems. Currently, MPC has a distinct advantage over other types of control methodologies in that it has the ability to take into account hard constraints on the system such as actuator position limits, actuator rate limits, and state or output limits such as the maximum possible normal acceleration of an aircraft. As mentioned earlier, the state space formulation of MPC employed in this research effort also has the useful feature of achieving guaranteed stability for all feasible optimizations based on the existence of a monotonically decreasing cost function.

1.2 Importance of Research

Because MPC is capable of handling system constraints such as actuator rate and position limits explicitly, it is of interest to employ it to study the problem of aircraft actuator failures. In the case of a damaged or failed control surface, quick action by the pilot may be required to maintain positive control of the aircraft. However, especially in the case of a statically unstable aircraft such as a modern fighter, certain circumstances may arise in which the pilot is incapable of controlling the aircraft subsequent to a control surface failure. At this point, the controller must either partially or totally intercede to prevent a catastrophic failure and loss of the aircraft.

Given sensor inputs from the control surfaces to determine the existence of a failure, an MPC controller will automatically redistribute control power in the attempt to both maintain stability and track the given setpoint. It should be noted that multiple

control surface failures or severe out-of-trim failures of certain control surfaces may prevent stabilization of the aircraft. Nevertheless, depending upon which control surface fails and the severity of the failure, an MPC controller can not only maintain stability, but also maintain near-nominal system performance as well.

The advantages of using a fault-tolerant controller are thus clear. In the case of a severe out-of-trim failure of a critical control element (e.g. stabilator, thrust vectoring nozzle, etc.), such a controller will often be able to stabilize the aircraft, leading to the preservation of both the aircraft and the pilot. For minor failures, a fault-tolerant controller should be able to maintain near-nominal performance.

1.3 Research Objectives

The primary objective of this thesis is to explore the fault tolerance capabilities of a state space formulation of MPC. Subsequent to a simulated control surface failure on the F-18 High Alpha Research Vehicle (HARV), the MPC controller will attempt to maintain stability and nominal performance while tracking a setpoint. Secondary objectives include studying the effect of data sample rate and MPC horizon lengths on system performance. These objectives will be accomplished using Matlab to develop the MPC controller and Simulink to demonstrate its effectiveness through simulation.

1.4 Thesis Overview

Chapter 2 addresses the essential elements of developing a state space formulation

of MPC. Coprime factorization, plant modification to accept incremental control inputs, a stabilizing inner feedback loop, prediction and constraint equations, and basic stability criteria for an MPC controller are discussed.

Chapter 3 contains information about the F-18 HARV and the methodology used in this research effort.

Chapter 4 presents the results of the Matlab simulations.

Chapter 5 offers conclusions based on the results of the simulations and ideas for further avenues of research in the area of fault-tolerant control using MPC.

2.0 Review of Literature

2.1 Historical Development of MPC

This section presents the historical foundations of the state space MPC scheme employed in this research effort. Generalized Predictive Control and Stable Generalized Predictive Control are briefly discussed, with emphasis placed on their respective advantages and disadvantages.

2.1.1 Generalized Predictive Control

The fundamental concept of Generalized Predictive Control (GPC) [1] is to use an explicit plant model to predict the plant outputs, y, along a prediction horizon, N, based on inputs implemented over a control horizon, N_u, and then minimize the expected sum of the Euclidean norm of plant output deviations from a setpoint trajectory, s, and the weighted Euclidean norm of control activity by finding an optimal set of incremental control inputs, Δu. The Single-Input Single-Output (SISO) expectation cost function representing this process is given by

$$J(k) = E\left\{\sum_{l=1}^{N}[y(k+l) - s(k+l)]^2 + \lambda\sum_{l=1}^{N_u}\Delta u(k+l-1)^2\right\} \qquad \textbf{(2.1)}$$

where λ is a weight applied to the control usage. An expectation operator is used because the plant model contains a noise term colored by a user-specified polynomial.

GPC has several advantages over other forms of controllers. It does not require prior knowledge of the closed loop poles, nor does it suffer ill effects from singularities due to closely-spaced poles and zeros [2]. Additionally, GPC allows for the incorporation of both input and output constraints in the control algorithm and provides for a wide variety of tuning parameters that are useful in satisfying the given performance objectives. Unfortunately, traditional GPC has the deficiency of having neither a general guaranteed stability result nor a general guaranteed robust stability result.

2.1.2 Stable Generalized Predictive Control

Stable Generalized Predictive Control (SGPC) retains the advantages of GPC, but adds guaranteed nominal stability through the creation of a stabilizing inner feedback loop based on the Youla parameterization of all stabilizing controllers [2]. The closed-loop poles are chosen such that Finite Impulse Response (FIR) behavior is achieved. FIR behavior implies that the system will achieve a steady state value over a finite horizon and is useful in that, when used in conjunction with a cost function that minimizes tracking error, it can guarantee stability and asymptotic tracking through the existence of a monotonically decreasing cost function given specific conditions on the horizons are satisfied [3]. The MPC optimization is then wrapped around the inner stabilizing feedback loop, and because the output of the MPC optimizer no longer feeds directly into the plant, the cost function must now be minimized with respect to a "quasi-reference" signal to be described in Section 2.2.2.

2.2 Mathematical Preliminaries for State Space MPC

In this section, the mathematical background information necessary to develop an MPC controller is presented. This information includes right and left coprime factorization, stabilizing inner feedback loops based on the Youla parameterization of all stabilizing controllers, and plant modification to accept control increments.

2.2.1 Coprime Factorization

Although the development in [2] is in terms of a SISO plant, the theory can be expanded to address the case of a Multi-Input Multi-Output (MIMO) system, as is shown in [4]. Consider the case of a discrete time plant $G_p(z)$ with ξ inputs and η outputs. For any proper transfer function matrix with both a stabilizable and a detectable realization, it is always possible to describe the plant in terms of a doubly coprime factorization [5]

$$\begin{aligned} G_p(z) &= N_p(z)\, M_p^{-1}(z) \\ &= \tilde{M}_p^{-1}(z)\, \tilde{N}_p(z) \end{aligned} \tag{2.2}$$

where $M_p(z)$, $N_p(z)$, $\tilde{M}_p(z)$, $\tilde{N}_p(z) \in \mathcal{RH}_\infty$ and $M_p(z)$, $\tilde{M}_p(z)$ are square and non-singular. For the purpose of introducing integral action into the system, it is convenient to modify the plant so that it operates on incremental changes in the control signal instead of the absolute control signal. To do so, provided that $G_p(z)$ does not have any zeros at $z = +1$, the differencing operator $\Delta = 1 - z^{-1}$ is employed, and a modified plant $G(z)$ is defined:

$$\begin{aligned} G(z) &= \frac{1}{\Delta} G_p(z) \\ &= N(z) M^{-1}(z) \\ &= \tilde{M}^{-1}(z) \tilde{N}(z) \end{aligned} \tag{2.3}$$

Additionally, we shall assign $M(z)$ and $N(z)$ to be

$$\begin{aligned} M(z) &= \Delta M_p(z) \\ N(z) &= N_p(z) \end{aligned} \tag{2.4}$$

where $M(z)$, $N(z)$, $\tilde{M}(z)$, $\tilde{N}(z) \in \mathcal{RH}_\infty$, and both pairs are coprime.

From the definition of coprimeness [5], two matrices $M(z)$, $N(z) \in \mathcal{RH}_\infty$ are right coprime if they have the same number of columns and there exist two matrices $\tilde{X}(z)$, $\tilde{Y}(z) \in \mathcal{RH}_\infty$ having the relation

$$\begin{bmatrix} \tilde{X} & \tilde{Y} \end{bmatrix} \begin{bmatrix} N \\ M \end{bmatrix} = \tilde{X}N + \tilde{Y}M = I \tag{2.5}$$

Two matrices $\tilde{M}(z)$, $\tilde{N}(z) \in \mathcal{RH}_\infty$ are left coprime if they have the same number of rows and there exist two matrices $X(z)$, $Y(z) \in \mathcal{RH}_\infty$ having the relation

$$\begin{bmatrix} \tilde{N} & \tilde{M} \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = \tilde{N}X + \tilde{M}Y = I \tag{2.6}$$

For any proper transfer function, it is possible to find both the left and right coprime factorizations simultaneously through the solution of the generalized Bezout identity:

$$\begin{bmatrix} \tilde{X} & -\tilde{Y} \\ \tilde{M} & \tilde{N} \end{bmatrix} \begin{bmatrix} N & Y \\ -M & X \end{bmatrix} = I \tag{2.7}$$

In a state space context, the right coprime factorization is achieved through the introduction of a state feedback variable, v. Consider a strictly proper discrete time plant $G(z)$ that acts on incremental changes in the control input, $\Delta u(k)$,

$$G(z) = \left[\begin{array}{c|c} A & B \\ \hline C & 0 \end{array}\right] \tag{2.8}$$

with (A,B) stabilizable and (C,A) detectable. The state space representation is

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u(k) \\ y(k) &= Cx(k) \end{aligned} \tag{2.9}$$

Now, introduce the state feedback variable, $v(k)$, of the form

$$v(k) = Z_r^{-1}\left[\, \Delta u(k) - F_r x(k) \,\right] \tag{2.10}$$

where F_r is chosen such that the eigenvalues of $A + BF_r$ all lie within the unit disk. By the assumption of stabilizability, this can always be accomplished. Z_r is chosen to satisfy a relation defined in Section 2.3.2. Substituting equation (2.10) into equation (2.9) yields

$$\begin{aligned} x(k+1) &= (A + BF_r)\,x(k) + BZ_r v(k) \\ \Delta u(k) &= F_r x(k) + Z_r v(k) \\ y(k) &= Cx(k) \end{aligned} \tag{2.11}$$

The transfer matrix from $v(k)$ to $\Delta u(k)$ is thus seen to be represented by

$$M(z) = \left[\begin{array}{c|c} (A + BF_r) & BZ_r \\ \hline F_r & Z_r \end{array}\right] \tag{2.12}$$

and the transfer matrix from $v(k)$ to $y(k)$ is

$$N(z) = \left[\begin{array}{c|c} (A + BF_r) & BZ_r \\ \hline C & 0 \end{array}\right] \tag{2.13}$$

Therefore, the relations between the state feedback variable and the input and output are

$$\begin{aligned} \Delta u(z) &= M(z)\,v(z) \\ y(z) &= N(z)\,v(z) \\ y(z) &= N(z)\,M^{-1}(z)\,\Delta u(z) \end{aligned} \tag{2.14}$$

Using the dual procedure to the one outlined previously, the left coprime factorization of $G(z)$ is given by:

$$\left[\begin{array}{cc} \tilde{M} & \tilde{N} \end{array}\right] := \left[\begin{array}{c|cc} A + LC & L & B \\ \hline Z_l C & Z_l & 0 \end{array}\right] \tag{2.15}$$

with a choice of L such that all the eigenvalues of $A + LC$ lie within the unit disk.

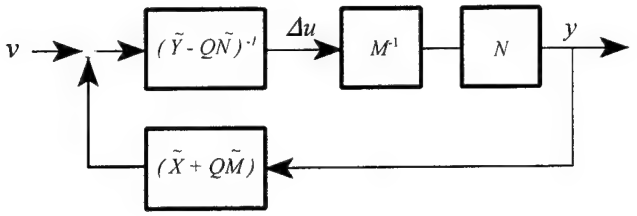


Figure 2.1 Stabilizing Inner Feedback Loop

2.2.2 Stabilizing Inner Feedback Loop

Given the assumption that $G(z)$ can be described by a coprime factorization, the transfer function matrices $X(z)$, $Y(z)$, $\tilde{X}(z)$, $\tilde{Y}(z) \in \mathcal{RH}_\infty$ exist and have the relation shown in equation (2.7). Solving the Bezout identity for a coprime factorization as in equation (2.7) then leads to the set of all stabilizing rational feedback controllers for $G(z)$, which are parameterized by

$$K = -(\tilde{Y} - Q\tilde{N})^{-1}(\tilde{X} + Q\tilde{M}) \tag{2.16}$$

for any $Q \in \mathcal{RH}_\infty$ [5]. Figure 2.1 depicts the setup of the stabilizing inner feedback controller for a MIMO system, where the "quasi-reference" signal $v(k) \in \mathbb{R}^\xi$ is the output of the MPC optimization routine, $y(k) \in \mathbb{R}^\eta$ is the system output, and $\Delta u(k) \in \mathbb{R}^\xi$ is the incremental control input such that $\Delta u(k) = u(k) - u(k\text{-}1)$. The particular construction of this feedback loop leads to the recovery of the relations in equation (2.14), the proof of which may be found in [4] and will not be repeated here. One will note that the transfer

function matrix Q does not appear in the recovery of equation (2.14) and may therefore be assumed to be zero. Although this feedback controller is implementable with any choice of $M(z)$ and $N(z)$, choosing them to be FIR operators leads to a FIR relationship between $y(z)$ and $\Delta u(z)$ and the "quasi-reference" signal, $v(z)$.

2.2.3 Plant Modification to Accept Control Increments

Consider the following discrete time state space realization:

$$\begin{aligned} x(k+1) &= Ax(k) + B_u u(k) + B_d d(k) \\ y(k) &= Cx(k) + w(k-1) + n(k) \end{aligned} \tag{2.17}$$

where $x(k) \in \mathbb{R}^{\zeta}$ is the state vector, $u(k) \in \mathbb{R}^{\xi}$ is the input vector, $d(k) \in \mathbb{R}^{\nu}$ is a state disturbance that may include both measured and unmeasured terms, $y(k) \in \mathbb{R}^{\eta}$ is the measured output, $w(k) \in \mathbb{R}^{\rho}$ is an output disturbance vector, and $n(k)$ represents a corruption of the output vector due to noise. In this form, however, the system is not useful for MPC in our context here because it operates on the absolute control signal, $u(k)$, and not the incremental change in the control signal, $\Delta u(k)$. To remedy this situation, we may augment the plant with a differencer, which is the equivalent of adding ξ integrators at the plant input. This method is not useful in the case of non-square systems, however, and will therefore not be utilized in this research effort. A more versatile method of modifying the plant, which is equivalent to adding a bank of integrators at either the plant input or output, may be expressed as

$$\begin{bmatrix} \Delta x(k+1) \\ y_x(k+1) \end{bmatrix} = \hat{A}\begin{bmatrix} \Delta x(k) \\ y_x(k) \end{bmatrix} + \hat{B}_u \Delta u(k) + \hat{B}_d \Delta d(k) + \hat{B}_w \Delta w(k)$$
$$y(k) = \hat{C}\begin{bmatrix} \Delta x(k) \\ y_x(k) \end{bmatrix} + n(k) \tag{2.18}$$

where

$$\hat{A} = \begin{bmatrix} A & 0 \\ CA & I \end{bmatrix}, \quad \hat{B}_u = \begin{bmatrix} B_u \\ CB_u \end{bmatrix}, \quad \hat{B}_d = \begin{bmatrix} B_d \\ CB_d \end{bmatrix}$$
$$\hat{B}_w = \begin{bmatrix} 0 \\ I \end{bmatrix}, \quad \hat{C} = \begin{bmatrix} 0 & I \end{bmatrix} \tag{2.19}$$

and where $\Delta x(k) = x(k) - x(k-1)$, $\Delta u(k) = u(k) - u(k-1)$, $\Delta d(k) = d(k) - d(k-1)$, $\Delta w(k) = w(k) - w(k-1)$ are the change in the state at time k, the change in the control input at time *k*, the change in the state disturbance vector at time *k*, and the change in the output disturbance at time *k*, respectively. y_x is the vector of outputs uncorrupted by measurement noise.

2.3 State Space Formulation of Constrained MPC

This section presents the development of a state space formulation of constrained MPC. First, the MPC horizons will be defined, and their relation to internal stability for a system operating with output feedback will be introduced. Next, the method for

developing an internally stabilizing feedback loop that produces a FIR relationship between the output of the MPC optimization and the plant input and output will be presented. Then, state and input prediction, input and output constraint formulation for a quadratic optimization, and system stability will be discussed. Lastly, this material will be integrated to implement a quadratic optimization MPC controller.

2.3.1 Definition of MPC Horizons

As a finite horizon control methodology, state space MPC requires the introduction of three independent horizons in order to function: the prediction horizon, the control horizon, and the optimization horizon. The prediction horizon is of length p and defines the number of time steps into the future for which the plant state or output vector is predicted at any given time, k, during the optimization process. The control horizon, q, defines the number of time steps over which incremental control input moves are available starting at time k, after which $\Delta u(k+l) = 0$ for $l \geq q$. The optimization horizon, r, defines the number of state feedback signals, $v(k+l)$, that are calculated by the optimizer. After r steps, $v(k+l)$ reaches the steady state value v^{∞} that drives the system to its setpoint, s^{∞}. The relative lengths of the various horizons are important because the stability of the system being controlled by the MPC process is directly related to the choice of these three horizons [4]. For a system using output feedback and an estimator for state prediction, stability of the closed loop system operating under constraints is only guaranteed if: $N - 2n \leq r \leq \min(p-2n, q-2n)$, where n is the order of the plant and N is the fewest number of steps required to drive the system to its steady-state setpoint, s^{∞}.

2.3.2 Internally Stabilizing Feedback Controller

Let the discrete time state space representation for a plant $G(z) \in \mathcal{RL}_\infty$ be given by

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u(k) \\ y(k) &= Cx(k) \end{aligned} \tag{2.20}$$

with (A,B) controllable and (C,A) observable, and where $x(k) \in \mathbb{R}^n$ is the system state vector. In order to implement the stabilizing inner feedback controller described in Section 2.2.2, we need to find the coprime factorization of (2.20). The right coprime factorization may be accomplished through the introduction of the state feedback variable, $v(k)$, in the form of equation (2.10). F_r is chosen such that the poles of $A + BF_r$ are at the origin, which leads to the recovery of the FIR relationship between the "quasi-reference" input, $v(k)$, and the plant input and output. These FIR relationships allow for the calculation of the far future value of $v(k)$ that causes the estimated plant output to reach its setpoint within the prediction horizon. The factors necessary to construct the stabilizing controller shown in Figure 2.1 are given in equations (2.15) and by

$$\begin{bmatrix} \tilde{Y} & \tilde{X} \end{bmatrix} := \left[\begin{array}{c|cc} A + LC & -B & L \\ \hline Z_l^{-1}F_r & Z_l^{-1} & 0 \end{array}\right] \tag{2.21}$$

Furthermore, the condition that $Z_l Z_r^{-1} = I$ is stipulated in order to attain the closed loop relations of equation (2.14).

2.3.3 State Prediction

In order to find the optimal set of reference inputs, $v(k+l)$, to attain and track the setpoint, it is first necessary to construct a methodology to determine the future system states based on these inputs. As most physical systems have at least some unmeasurable states, it is convenient to base the prediction equations on an estimator of the form

$$\begin{aligned} \hat{x}(k+1) &= A\hat{x}(k) + B\Delta u(k) + L[C\hat{x}(k) - y(k)] \\ \Delta u(k) &= F_r\hat{x}(k) + Z_r v(k) \end{aligned} \tag{2.22}$$

with $\hat{x}(k) \in \mathbb{R}^{\kappa}$. Substituting for $\Delta u(k)$ in the first equation in set (2.22) then leads to

$$\hat{x}(k+1) = (A + BF_r + LC)\hat{x}(k) + BZ_r v(k) + Ly(k) \tag{2.23}$$

Using equation (2.23), it is now possible to express the state estimate for a particular point l on the prediction horizon as

$$\begin{aligned} \hat{x}(k+l+1) &= F(l)\hat{x}(k+l) + Gv(k+l) + H(l)y(k) \\ l &= 0 \ldots p-1 \end{aligned} \tag{2.24}$$

where $F(0) = A + BF_r + LC$ and $F(1) \ldots F(p\text{-}1) = A + BF_r$, $G = BZ_r$, $H(0) = L$ and $H(1) \ldots H(p\text{-}1) = 0$. The terms dealing with the output feedback are eliminated for all times greater than k because the actual system output is not available at any future time.

Utilizing equation (2.24), it is possible to construct a vector of future state estimates using the notation defined in [4] and [6], but it is first necessary to define the

vectors

$$\begin{aligned} \tilde{v}(k) &= \left[\, v(k)^T \cdots v(k+r-1)^T \,\right]^T \\ \tilde{v}^{\infty}(k) &= \left[\, v(k+r)^T \cdots v(k+\rho-1)^T \,\right]^T \\ \Delta\tilde{u}(k) &= \left[\, \Delta u(k)^T \cdots \Delta u(k+q-1)^T \,\right]^T \\ \tilde{u}(k) &= \left[\, u(k)^T \cdots u(k+q-1)^T \,\right]^T \end{aligned} \tag{2.25}$$

where $\rho = \max(p, q)$. The vector of state estimate predictions can then be written as

$$\begin{bmatrix} \hat{x}(k+1) \\ \vdots \\ \hat{x}(k+p) \end{bmatrix} = \tilde{F}\hat{x}(k) + \tilde{G}\tilde{v}(k) + \tilde{G}^{\infty}\tilde{v}^{\infty}(k) + \tilde{H}y(k) \tag{2.26}$$

in which $\tilde{F}$, $\tilde{G}$, $\tilde{G}^{\infty}$, and $\tilde{H}$ are matrix functions of $F(l)$, G, and $H(l)$. These matrix functions can be represented using the notation [4,6]

$$\prod_{j=m}^{n} F(j) = \begin{cases} F(m)\,F(m-1)\ldots F(n) & m > n \\ F(m) & m = n \\ I & m = n-1 \\ 0 & otherwise \end{cases} \tag{2.27}$$

which allows the predicted state estimate at each time step to be defined by

$$\hat{x}(k+l+1) = \prod_{j=l}^{0} F(j)\,\hat{x}(k) + \sum_{i=0}^{\rho-1} \left\{ \left[\prod_{j=l}^{i+1} F(j) \right] G v(k+i) \right\} + \left[\prod_{j=l}^{1} F(j) \right] H(0) y(k) \tag{2.28}$$

Using equation (2.28), it is thus possible to formulate the state estimate prediction matrices $\tilde{F}$, $\tilde{G}$, $\tilde{G}^{\infty}$, and $\tilde{H}$:

$$\tilde{F} = \left.\begin{bmatrix} F(0) \\ F(1)F(0) \\ \vdots \\ \prod_{j=p-1}^{0} F(j) \end{bmatrix}\right\} p$$

$$\tilde{G} = \left.\begin{bmatrix} G & 0 & \cdots & 0 \\ F(1)G & G & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \left[\prod_{j=r}^{1} F(j)\right]G & \left[\prod_{j=r}^{2} F(j)\right]G & \vdots & G \\ \vdots & \vdots & \ddots & \vdots \\ \left[\prod_{j=p-1}^{1} F(j)\right]G & \left[\prod_{j=p-1}^{2} F(j)\right]G & \cdots & \left[\prod_{j=p-1}^{r} F(j)\right]G \end{bmatrix}\right\} p \qquad \textbf{(2.29)}$$

$$\tilde{G}^{\infty} = \left.\begin{bmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ G & 0 & \cdots & 0 \\ F(r+1)G & G & \cdots & 0 \\ \left[\prod_{j=r+2}^{r+1} F(j)\right]G & F(r+2)G & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \left[\prod_{j=p-1}^{r+1} F(j)\right]G & \left[\prod_{j=p-1}^{r+2} F(j)\right]G & \cdots & \left[\prod_{j=p-1}^{\rho} F(j)\right]G \end{bmatrix}\right\} p$$

$$\tilde{H} = \left.\begin{bmatrix} H(0) \\ F(1)H(0) \\ \vdots \\ \left[\prod_{j=p-1}^{1} F(j)\right]H(0) \end{bmatrix}\right\} p$$

Similarly, it is possible to construct a vector of predicted future incremental control input moves, $\Delta\tilde{u}(k)$

$$\Delta\tilde{u}(k) = \tilde{F}_{\Delta}\hat{x}(k) + \tilde{G}_{\Delta}\tilde{v}(k) + \tilde{G}_{\Delta}^{\infty}\tilde{v}^{\infty} + \tilde{H}_{\Delta}y(k) \tag{2.30}$$

where $\tilde{F}_{\Delta}$, $\tilde{G}_{\Delta}$, $\tilde{G}_{\Delta}^{\infty}$, and $\tilde{H}_{\Delta}$ are matrix functions of F_r, Z_r, $F(l)$, G, and $H(l)$. To do so, we must first introduce a modified form of the controller input from equation (2.22) that is defined for any time l along the control horizon:

$$\begin{aligned} \Delta u(k+l) &= F_r\hat{x}(k+l) + Z_r v(k+l) \\ l &= 0 \ldots q-1 \end{aligned} \tag{2.31}$$

Inserting the state estimate prediction equations from (2.24) into (2.31) then yields an expression for the vector of future incremental control inputs

$$\begin{aligned} \Delta u(k+l) = F_r \prod_{j=l-1}^{0} F(j)\hat{x}(k) + F_r \sum_{i=0}^{\rho-1} \left\{ \left[\prod_{j=l-1}^{i+1} F(j) \right] G v(k+i) \right\} + \\ Z_r v(k+l) + F_r \left[\prod_{j=l-1}^{1} F(j) \right] H(0) y(k) \end{aligned} \tag{2.32}$$

from which we may derive $\tilde{F}_{\Delta}$, $\tilde{G}_{\Delta}$, $\tilde{G}_{\Delta}^{\infty}$, and $\tilde{H}_{\Delta}$:

$$\tilde{F}_{\Delta} = \left.\begin{bmatrix} F_r \\ F_r F(0) \\ \vdots \\ F_r \left[\prod_{j=q-2}^{0} F(j) \right] \end{bmatrix}\right\} q$$

$$\tilde{G}_{\Delta} = \left.\begin{bmatrix} Z_r & 0 & \cdots & 0 \\ F_r G & Z_r & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ F_r\left[\prod_{j=r-2}^{1} F(j)\right]G & F_r\left[\prod_{j=r-2}^{2} F(j)\right]G & \cdots & Z_r \\ F_r\left[\prod_{j=r-1}^{1} F(j)\right]G & F_r\left[\prod_{j=r-1}^{2} F(j)\right]G & \cdots & F_r G \\ \vdots & \vdots & \ddots & \vdots \\ F_r\left[\prod_{j=q-2}^{1} F(j)\right]G & F_r\left[\prod_{j=q-2}^{2} F(j)\right]G & \cdots & F_r\left[\prod_{j=q-2}^{r} F(j)\right]G \end{bmatrix}\right\} q$$

$$\tilde{G}_{\Delta}^{\infty} = \left.\begin{bmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ Z_r & 0 & \cdots & 0 \\ F_r G & Z_r & \cdots & 0 \\ F_r F(r+1) G & F_r G & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ F_r\left[\prod_{j=q-2}^{r+1} F(j)\right]G & F_r\left[\prod_{j=q-2}^{r+2} F(j)\right]G & \cdots & F_r\left[\prod_{j=q-2}^{\rho} F(j)\right]G \end{bmatrix}\right\} q \qquad \textbf{(2.33)}$$

$$\tilde{H}_{\Delta} = \left.\begin{bmatrix} 0 \\ F_r H(0) \\ \vdots \\ \left[\prod_{j=q-2}^{1} F(j)\right]H(0) \end{bmatrix}\right\} q$$

Lastly, it is possible to derive an expression for the absolute control input at any point l along the control horizon based on the fact that $u(k+l) = u(k-1) + \sum_{j=0}^{l} \Delta u(k+j)$:

$$u(k+l) = \sum_{h=0}^{l} \left\{ F_r \left[\prod_{j=h-1}^{0} F(h) \right] \hat{x}(k) + F_r \sum_{i=0}^{\rho-1} \left(\left[\prod_{j=h-1}^{i+1} F(j) \right] G v(k+i) \right) + Z_r v(k+h) + F_r \left[\prod_{j=h-1}^{1} F(j) \right] H(0) y(k) \right\} + u(k-1) \tag{2.34}$$

The vector of predicted absolute control inputs can thus be represented by

$$\tilde{u}(k) = \tilde{F}_u \hat{x}(k) + \tilde{G}_u \tilde{v}(k) + \tilde{G}_u^{\infty} \tilde{v}^{\infty} + \tilde{H}_u y(k) + \tilde{I} u(k-1) \tag{2.35}$$

where $\tilde{F}_u$, $\tilde{G}_u$, $\tilde{G}_u^{\infty}$, and $\tilde{H}_u$ are matrix functions of F_r, Z_r, $F(l)$, G, and $H(l)$, and $\tilde{I}$ is a column of ξ x ξ identity matrices:

$$\tilde{F}_u = \left\{ \begin{bmatrix} F_r \\ F_r[F(0) + I] \\ \vdots \\ F_r \sum_{i=0}^{q-1} \left[\prod_{j=i-1}^{0} F(i) \right] \end{bmatrix} \right\} q$$

$$\tilde{G}_u = \begin{bmatrix} Z_r & 0 & \cdots & 0 \\ F_r G + Z_r & Z_r & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=0}^{r} F(r)\left[\prod_{j=i-1}^{1} F(j)\right] G + Zr & \sum_{i=0}^{r} F(r)\left[\prod_{j=i-1}^{2} F(j)\right] G + Zr & \cdots & F_r G + Z_r \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=0}^{q-1} F(r)\left[\prod_{j=i-1}^{1} F(j)\right] G + Zr & \sum_{i=0}^{q-1} F(r)\left[\prod_{j=i-1}^{2} F(j)\right] G + Zr & \cdots & \sum_{i=0}^{q-1} F(r)\left[\prod_{j=i-1}^{r} F(j)\right] G + Zr \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \\ \\ \end{matrix} \left.\begin{matrix} \\ \\ \\ \\ \\ \end{matrix}\right\} q \qquad \textbf{(2.37)}$$

$$\tilde{G}_u^{\infty} = \begin{bmatrix} 0 & 0 \\ \vdots & \vdots \\ Z_r & 0 \\ F_r G + Z_r & Z_r \\ \sum_{i=0}^{r+1} F(r) \left[\prod_{j=i-1}^{r+1} F(j) \right] G + Zr & F_r G + Z_r \\ \vdots & \vdots \\ \sum_{i=0}^{q-1} F(r) \left[\prod_{j=i-1}^{r+1} F(j) \right] G + Zr & \sum_{i=0}^{q-1} F(r) \left[\prod_{j=i-1}^{r+2} F(j) \right] G + Zr \end{bmatrix}$$

$$\left.\begin{bmatrix} \cdots & 0 \\ \ddots & \vdots \\ \cdots & 0 \\ \cdots & 0 \\ \ddots & 0 \\ \cdots & \sum_{i=0}^{q-1} F(r) \left[\prod_{j=i-1}^{\rho} F(j) \right] G + Zr \end{bmatrix}\right\} q$$

$$\tilde{H}_u = \left.\begin{bmatrix} 0 \\ F_r H(0) \\ \vdots \\ \sum_{i=0}^{q-1} F_r \left[\prod_{j=i-1}^{1} F(j) \right] H(0) \end{bmatrix}\right\} q$$

$$\tilde{I} = \begin{bmatrix} I_\xi & I_\xi & \cdots & I_\xi \end{bmatrix}^T$$

2.3.4 System Constraints

For any physical system, there are limits to the rate at which a control input may be applied and to the magnitude of the absolute control input. Additionally, all physical systems have hard or soft constraints that may be represented either by a limitation on a specific state or by a limitation on a linear combination of states. An example of a hard constraint would be the normal acceleration beyond which airframe structural failure is imminent. A soft constraint might be the normal acceleration that the pilot is expected to be able to withstand during Air Combat Maneuvering. The rate, position, and state estimate constraints do not necessarily have the same upper and lower limits either, so they are typically expressed as having minimum and maximum bounds:

$$\begin{aligned} \Delta u_{\min}(k+i) \le \Delta u(k+i) \le \Delta u_{\max}(k+i) \quad & i = 0 \ldots q-1 \\ u_{\min}(k+i) \le u(k+i) \le u_{\max}(k+i) \quad & i = 0 \ldots q-1 \\ \hat{x}_{\min}(k+i) \le \hat{x}(k+i) \le \hat{x}_{\max}(k+i) \quad & i = 1 \ldots p \end{aligned} \tag{2.37}$$

These bounds may be held constant throughout the control and prediction horizons, or they may be chosen to vary with time. Additionally, it is possible to introduce output constraints on the system by limiting linear combinations of the state estimate constraints. Furthermore, if one wishes to consider a particular control increment, control input, or state to be unconstrained, the maximum or minimum limit may be set to $+\infty$ or $-\infty$, respectively.

The use of the Matlab *qp* command for the constrained quadratic optimization process to determine the series of "quasi-reference" signals, $\tilde{v}(k)$, at each time step makes

it necessary to formulate the rate, position, and state constraints on the system as a matrix inequality of the form

$$\tilde{D}\tilde{v}(k) \leq \tilde{E}c(k)$$

where **(2.38)**

$$c(k) = \left[\hat{x}(k)^T \;\; y(k)^T \;\; \tilde{v}^{\infty}(k)^T \;\; u(k-1)^T \;\; l(k)^T \;\; m(k)^T \;\; n(k)^T \right]^T$$

and where $l(k) \in \mathbb{R}^{\lambda}$, $m(k) \in \mathbb{R}^{\mu}$, and $n(k) \in \mathbb{R}^{\nu}$ represent vectors of constraints on the control rates, absolute control inputs, and state estimates, respectively. Additionally, λ is the number of constraints on input rates across the control horizon, μ is the number of input constraints across the control horizon, and ν is the number of state estimate constraints across the prediction horizon. The elementwise inequalities needed to obtain (2.38) are

$$\begin{aligned} L_{\lambda}\Delta\tilde{u}(k) &\leq l(k) \\ M_{\mu}\tilde{u}(k) &\leq m(k) \\ N_{\nu}\begin{bmatrix} \hat{x}(k+1) \\ \vdots \\ \hat{x}(k+p) \end{bmatrix} &\leq n(k) \end{aligned} \qquad \textbf{(2.39)}$$

where in the case without linear combinations of constraints L_{λ}, M_{μ}, and N_{ν} are defined by

$$L_{\lambda} = \begin{bmatrix} -I_{q\xi} \\ I_{q\xi} \end{bmatrix}, \quad M_{\mu} = \begin{bmatrix} -I_{q\xi} \\ I_{q\xi} \end{bmatrix}, \quad N_{\nu} = \begin{bmatrix} -I_{p\kappa} \\ I_{p\kappa} \end{bmatrix} \qquad \textbf{(2.40)}$$

and $l(k)$, $m(k)$, and $n(k)$ are defined by:

$$l(k) = \begin{bmatrix} -l_{\min}(k) \\ l_{\max}(k) \end{bmatrix}, \quad m(k) = \begin{bmatrix} -m_{\min}(k) \\ m_{\max}(k) \end{bmatrix}, \quad n(k) = \begin{bmatrix} -n_{\min}(k) \\ n_{\max}(k) \end{bmatrix} \tag{2.41}$$

where $l \in \mathbb{R}^{2q\xi}$, $m \in \mathbb{R}^{2q\xi}$, and $n \in \mathbb{R}^{2p\kappa}$. Here, λ and μ are both assigned to be $2q\xi$ because there are 2ξ upper and lower input constraints at each time step over the control horizon, q. Similarly, ν is set to $2p\kappa$ because there are 2κ upper and lower state estimate constraints at each time step over the prediction horizon, p. By stacking the upper and lower constraints as in equations (2.40) and (2.41) and substituting the prediction equations from (2.26), (2.30), and (2.35) into equation (2.39), it is possible to incorporate both the minimum and maximum bounds on the system into the form of equation (2.38), where

$$\tilde{D} = \begin{bmatrix} L_\lambda \tilde{G}_\Delta \\ M_\mu \tilde{G}_u \\ N_\nu \tilde{G} \end{bmatrix}, \quad \tilde{E} = \begin{bmatrix} -L_\lambda \tilde{F}_\Delta & -L_\lambda H_\Delta & -L_\lambda \tilde{G}_\Delta^\infty & 0 & I & 0 & 0 \\ -M_\mu \tilde{F}_u & -M_\mu \tilde{H}_u & -M_\mu \tilde{G}_u^\infty & -M_\mu \tilde{I} & 0 & I & 0 \\ -N_\nu \tilde{F} & -N_\nu \tilde{H} & -N_\nu \tilde{G}^\infty & 0 & 0 & 0 & I \end{bmatrix} \tag{2.42}$$

2.3.5 Quadratic Programming Implementation

In this formulation of state space predictive control, the goal is to find the optimal series of "quasi-reference" signals, $v(k+l)$, such that the cost function

$$J(k) = \sum_{l=1}^{p} \left\| \hat{C}\hat{x}(k+l) - s(k+l) \right\|_{R_y}^2 + \sum_{l=1}^{q} \left\| \Delta u(k+l-1) \right\|_{R_u}^2$$
$$R_y > 0,\ R_u \geq 0 \qquad \textbf{(2.43)}$$

is minimized. R_y and R_u are weights applied to the tracking and control power terms of the cost function in order to adjust the relative penalties on setpoint tracking and control usage. Expanding this cost function and substituting in the prediction equations (2.26), (2.30), and (2.35), one may rewrite (2.43) as the equivalent cost function

$$J(k) = \tilde{v}(k)^T S\, \tilde{v}(k) + \left[\, \hat{x}(k)^T \;\; y(k)^T \;\; \tilde{v}^{\infty}(k)^T \;\; \tilde{s}(k)^T \,\right] T\, \tilde{v}(k) + \kappa \qquad \textbf{(2.44)}$$

where S is given by

$$S = \tilde{G}^T \tilde{C}^T \tilde{R}_y \tilde{C} \tilde{G} + \tilde{G}_{\Delta}^T \tilde{R}_u \tilde{G}_{\Delta} \qquad \textbf{(2.45)}$$

T is given by

$$T = 2\left[\, \tilde{C}\tilde{F} \;\; \tilde{C}\tilde{H} \;\; \tilde{C}\tilde{G}^{\infty} \;\; -I_{p\eta} \,\right]^T \tilde{R}_y \tilde{C} \tilde{G} + 2\left[\, \tilde{F}_{\Delta} \;\; \tilde{H}_{\Delta} \;\; \tilde{G}_{\Delta}^{\infty} \;\; 0 \,\right]^T \tilde{R}_u \tilde{G}_{\Delta} \qquad \textbf{(2.46)}$$

and those terms not involving $\tilde{v}(k)$ are grouped together in κ:

$$\begin{aligned} \kappa = {} & \left(\tilde{F}\hat{x}(k) + \tilde{G}^{\infty}\tilde{v}^{\infty} + \tilde{H}y(k)\right)^T \tilde{C}^T \tilde{R}_y \tilde{C} \left(\tilde{F}\hat{x}(k) + \tilde{G}^{\infty}\tilde{v}^{\infty} + \tilde{H}y(k)\right) + \\ & \left(\tilde{F}_{\Delta}\hat{x}(k) + \tilde{G}_{\Delta}^{\infty}\tilde{v}^{\infty} + \tilde{H}_{\Delta}y(k)\right)^T \tilde{R}_u \left(\tilde{F}_{\Delta}\hat{x}(k) + G_{\Delta}^{\infty}\tilde{v}^{\infty} + \tilde{H}_{\Delta}y(k)\right) + \\ & \tilde{s}(k)^T \tilde{R}_y \left[\, \tilde{s}(k) - 2\tilde{C}\left(\tilde{F}\hat{x}(k) + \tilde{H}y(k) + \tilde{G}^{\infty}\tilde{v}^{\infty}\right)\right] \end{aligned} \qquad \textbf{(2.47)}$$

Additionally, $\tilde{C} = \text{diag}(\hat{C},\ldots,\hat{C})$, $\tilde{R}_y = \text{diag}(R_y,\ldots,R_y)$, and $\tilde{R}_u = \text{diag}(R_u,\ldots,R_u)$. Furthermore, based on Lemma 5.1 in [4], it is possible to calculate the far future value of the "quasi-reference" signal, v^∞, from the system of equations

$$\left[C \sum_{j=1}^{2n} (A + BF_r)^{j-1} BZ_r \right] v^\infty = s^\infty \tag{2.48}$$

The MPC optimization may then be realized as the quadratic programming problem:

$$\min_{\tilde{v}(k)} \quad \tilde{v}(k)^T S \tilde{v}(k) + \left[\hat{x}(k)^T \;\; y(k)^T \;\; \tilde{v}^{\infty T} \;\; \tilde{s}(k)^T \right] T \tilde{v}(k) \tag{2.49}$$

subject to the constraints:

$$\tilde{D}\tilde{v}(k) \leq \tilde{E}c(k) \tag{2.50}$$

It should be emphasized at this point that this is not a static quadratic optimization problem. At each time step, the vector of state estimates, the plant output, the system constraints, and the vectors of steady state reference signals and setpoints are updated and a new constrained quadratic optimization is performed. These updates allow for dynamic setpoint and constraint changes, and the combination of the on-line optimization process and the stabilizing inner feedback loop is the mechanism by which system stability is maintained. The stable model predictive system is implementable based on Figure 2.2.

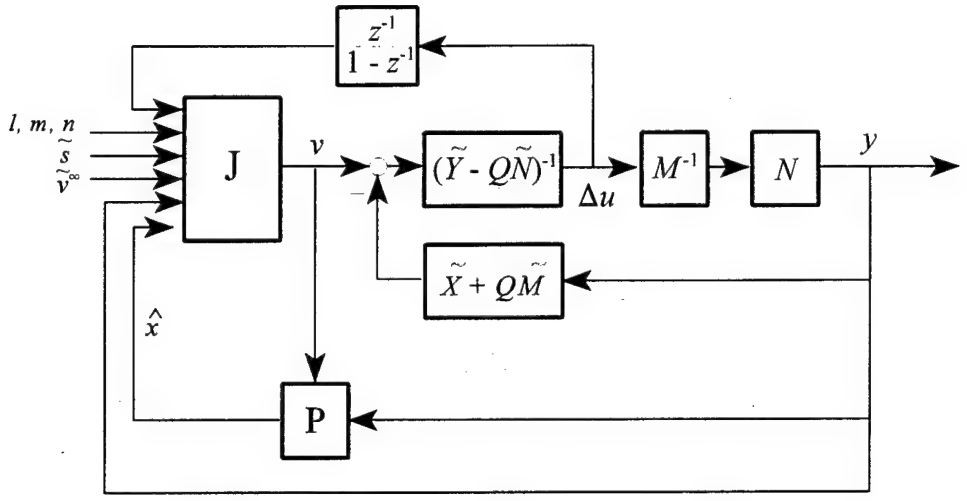


Figure 2. 2 Stable Model Predictive Control System

3.0 System and Maneuver Description and Model Formulation

3.1 F-18 High Alpha Research Vehicle (HARV)

The HARV is a NASA-owned modified version of the F-18 Hornet single-seat fighter/attack aircraft built by the McDonnell Aircraft Company, St. Louis, Missouri. It is powered by two General Electric F404-GE-400 afterburning turbofans, each capable of producing 16,000 pounds static thrust in afterburner at sea level. In order to augment yaw and pitch performance at high angles of attack, three externally-mounted thrust vectoring vanes were added to each engine to deflect the exhaust. The addition of these paddles necessitated the removal of the divergent portion of the engine nozzles, which prevents the HARV from achieving supersonic flight. However, afterburner operation is still possible. A comparison of the physical characteristics of a standard F-18 and the HARV is included in Table 3.1. A three-view drawing of the HARV may be found in Figure 3.1.

The most important functional improvement of the HARV over a standard F-18 is the addition of the thrust vectoring vane system. At low airspeed, conventional aerodynamic control surfaces are rendered ineffective because low dynamic pressure prevents them from generating the necessary moments to pitch, roll, and yaw an airplane. Furthermore, flight at high angles of attack produces cross-axis coupling of aerodynamic controls, which in turn complicates the development of a flight control system. One solution to both these difficulties is to use thrust vectoring. Thrust vectoring can be used symmetrically to produce either pitching moments or yawing moments, and it can be used

asymmetrically to produce rolling moments. Also, because the moments produced by thrust vectoring remain aligned with the aircraft axes, cross-axis coupling is avoided. Another advantage of incorporating thrust vectoring into an aircraft is control redundancy. In the event of an actuator failure on one of the aerodynamic control elements, the thrust vectoring system is capable of supplementing the moments produced by the functioning control surfaces.

Table 3.1: Physical Characteristics of the Unmodified and Modified F-18

Parameter	Unmodified	Modified
Weight, lb	31,980	36,099
Reference wing area, ft^2	400	400
Reference m.a.c., ft	11.52	11.52
Reference span, ft	37.4	37.4
Center of gravity	-	-
Percent m.a.c.	21.9	23.8
Fuselage ref. station	454.33	456.88
Waterline	105.24	105.35
Roll inertia, slug-ft^2	22,040	22,789
Pitch inertia, slug-ft^2	124,554	176,809
Yaw inertia, slug-ft^2	139,382	191,744
Product of inertia, slug-ft^2	-2,039	-2,305
Overall length, ft	56	56
Wing aspect ratio	3.5	3.5
Stabilator span, ft	21.6	21.6
Stabilator area, ft^2	88.26	86.48

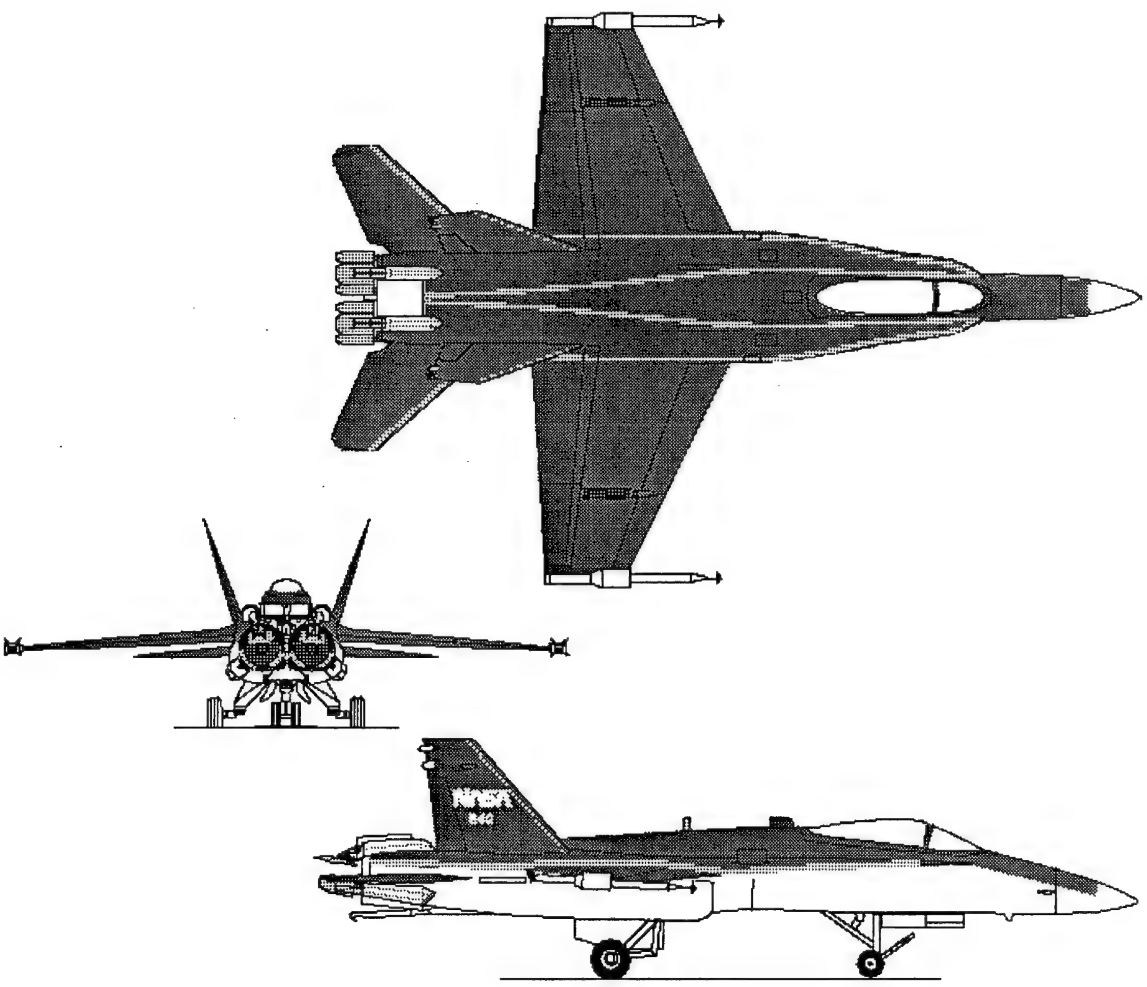


Figure 3.1 3-view drawing of the F-18 HARV

The F-18 HARV is equipped with five pairs of aerodynamic control surfaces:

(1) Independent stabilators capable of symmetric deflection for longitudinal maneuvering and differential deflection for aiding roll.

(2) Single-slotted ailerons with a ±25 degree range of operation.

(3) Leading-edge flaps that can deflect to a maximum of 33 degrees down. They can also operate differentially ±3 degrees for aiding roll.

(4) Single-slotted trailing-edge flaperons capable of a maximum 45 degree downward deflection in the landing configuration and a ±8 degree differential deflection. Additionally, these flaperons are scheduled with Mach number and angle of attack to decrease drag and augment stability.

(5) Twin rudders used for roll coordination and directional control.

The F-18 is also equipped with large leading edge extensions (LEX) that extend from the wing leading edge to just forward of the cockpit in order to provide enhanced maneuverability and lift production at high angles of attack. A complete listing of position and rate limits for the aerodynamic control surfaces is presented in Table 3.2.

Table 3.2: Aerodynamic Control Surfaces

Control Surface	Position Limit, deg	Rate Limit, deg/sec
Stabilator		40
TEU	24	
TED	10.5	
Aileron		100
TEU	25	
TED	45	
Rudder		82
TEL	30	
TER	30	
Trailing-edge flap		18
TEU	8	
TED	45	
Leading-edge flap		15
LEU	3	
LED	33	

The non-aerodynamic controls of the F-18 HARV are throttle position and thrust vectoring angle. Information on these control elements is provided in Table 3.3.

Table 3.3: Propulsive Control Elements	
Thrust Vectoring Vanes	
Dimension, in.	
Upper	20 by 20
Lower	20 by 15
Area, in^2	
Upper	358.76
Lower	263.64
Position Limit, deg	
Upper	-10
Lower	25
Rate Limit, deg/sec	80
Throttle	
Position Limit, deg	
Upper	127
Lower	54
Rate Limit, deg/sec	30

3.2 F-18 HARV Model

Summing forces and moments acting on an aircraft, the non-linear equations of motion can in general be represented by

$$\dot{x}(t) = f(x(t), u(t)) \tag{3.1}$$

where $x(t)$ is the vector of aircraft states and $u(t)$ is the vector of aircraft control inputs. Furthermore, at any arbitrary equilibrium point (x_o, u_o), the derivative of the state vector is

equal to the zero vector:

$$f(x_o, u_o) = 0 \tag{3.2}$$

Assuming small perturbations to the aircraft states and taking equation (3.2) into account, we can use a Taylor series to expand equation (3.1) and, neglecting higher order terms in the expansion, linearize the aircraft equations of motion about an equilibrium point. It is then possible to represent small motions about this equilibrium point with the state space system

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{3.3}$$

where $x(t)$ and $u(t)$ now represent perturbations in the states and inputs about their trim values.

The F-18 HARV state space models used in this research effort are taken from [7], but were originally provided by NASA Langley. These models represent both longitudinal and lateral-directional aircraft behavior about three trim conditions described in Section 3.4. In these state space systems, $A \in \mathbb{R}^{10 \times 10}$, $B \in \mathbb{R}^{10 \times 13}$, the vector of aircraft states is

$$x = \left[\begin{array}{cccccccccc} V_T & \alpha & \beta & p & q & r & \phi & \theta & \psi & h \end{array} \right]^T \tag{3.4}$$

and the vector of aircraft inputs is

$$u = \left[\begin{array}{cccccccccccc} \delta_{TVL} & \delta_{TVR} & \delta_{RL} & \delta_{RR} & \delta_{AL} & \delta_{AR} & \delta_{SL} & \delta_{SR} & \delta_{LEL} & \delta_{LER} & \delta_{TEL} & \delta_{TER} & \delta_T \end{array} \right]^T \tag{3.5}$$

Table 3.4 contains definitions of these quantities and their associated units.

Table 3.4: Definitions of F-18 HARV States and Control Inputs		
States:		
V_T	Perturbation in true airspeed	ft/sec
α	Perturbation in angle of attack	rad
β	Perturbation in sideslip angle	rad
p	Perturbation in roll rate	rad/sec
q	Perturbation in pitch rate	rad/sec
r	Perturbation in yaw rate	rad/sec
ϕ	Perturbation in roll angle	rad
θ	Perturbation in pitch angle	rad
ψ	Perturbaion in yaw angle	rad
h	Perturbation in altitude	ft
Controls:		
δ_{TVL}	Perturbation in left thrust vectoring vane deflection	deg
δ_{TVR}	Perturbation in right thrust vectoring vane deflection	deg
δ_{RL}	Perturbation in left rudder deflection	deg
δ_{RR}	Perturbation in right rudder deflection	deg
δ_{AL}	Perturbation in left aileron deflection	deg
δ_{AR}	Perturbation in right aileron deflection	deg
δ_{SL}	Perturbation in left stabilator deflection	deg
δ_{SR}	Perturbation in right stabilator deflection	deg
δ_{LEL}	Perturbation in left leading edge flap deflection	deg
δ_{LER}	Perturbation in right leading edge flap deflection	deg
δ_{TEL}	Perturbation in left trailing edge flap deflection	deg
δ_{TER}	Perturbation in right trailing edge flap deflection	deg
δ_T	Perturbation in throttle position	deg

For a conventional aircraft in which each control surface pair functions as a single entity, linearization of the aircraft equations of motion about a trim condition permits the assumption that the longitudinal and lateral-directional dynamics are decoupled. However, in the case of an aircraft such as the F-18 HARV, each control surface element is capable of operation independent of its counterpart and when deflected excites both longitudinal and lateral dynamics. In order to treat the longitudinal and lateral-directional dynamics of the HARV independently after linearization, it is necessary to decompose the motion of each pair of control elements into its symmetric and differential components. The symmetric component affects longitudinal dynamics, whereas the differential component produces lateral-directional motion. Define these components as

$$\begin{aligned} \delta_{XS} &= \delta_{XL} + \delta_{XR} \\ \delta_{XD} &= \delta_{XL} - \delta_{XR} \end{aligned} \tag{3.6}$$

where X represents an arbitrary control element, R and L specify the right or left control surface, D indicates differential deflection, and S indicates symmetric deflection. Using these definitions, it is then possible to formulate the decoupled state space system

$$\begin{bmatrix} \dot{x}_{LONG} \\ \dot{x}_{LAT} \end{bmatrix} = \begin{bmatrix} A_{LONG} & 0 \\ 0 & A_{LAT} \end{bmatrix} \begin{bmatrix} x_{LONG} \\ x_{LAT} \end{bmatrix} + \begin{bmatrix} B_{LONG} & 0 \\ 0 & B_{LAT} \end{bmatrix} \begin{bmatrix} u_{LONG} \\ u_{LAT} \end{bmatrix} \tag{3.7}$$

where $A_{LONG}, A_{LAT} \in \mathbb{R}^{5\times5}$, $B_{LONG} \in \mathbb{R}^{5\times6}$, $B_{LAT} \in \mathbb{R}^{5\times7}$, the longitudinal state vector and input vector are

$$x_{LONG} = \begin{bmatrix} V_T & \alpha & q & \theta & h \end{bmatrix}^T \qquad u_{LONG} = \begin{bmatrix} \delta_{TVS} & \delta_{AS} & \delta_{SS} & \delta_{LES} & \delta_{TES} & \delta_T \end{bmatrix}^T \quad \textbf{(3.8)}$$

and the lateral-directional state vector and input vector are

$$x_{LAT} = \begin{bmatrix} \beta & p & r & \phi & \psi \end{bmatrix}^T \qquad u_{LAT} = \begin{bmatrix} \delta_{TVD} & \delta_{AD} & \delta_{SD} & \delta_{LED} & \delta_{TED} & \delta_{RD} & \delta_{RS} \end{bmatrix}^T \quad \textbf{(3.9)}$$

Only longitudinal dynamics are considered in this thesis, so extracting the longitudinal portion of the plant and input matrices of the system of equation (3.7) and ignoring the redundant state, h, yields

$$\begin{aligned} \dot{x}_{LONG}(t) &= Ax_{LONG}(t) + Bu_{LONG}(t) \\ y(t) &= Cx_{LONG}(t) \end{aligned} \qquad \textbf{(3.10)}$$

where $A \in \mathbb{R}^{4\times4}$, $B \in \mathbb{R}^{4\times6}$, $C \in \mathbb{R}^{3\times4}$, the state and input vectors are

$$x_{LONG} = \begin{bmatrix} V_T & \alpha & q & \theta \end{bmatrix}^T \qquad u_{LONG} = \begin{bmatrix} \delta_{TVS} & \delta_{AS} & \delta_{SS} & \delta_{LES} & \delta_{TES} & \delta_T \end{bmatrix} \quad \textbf{(3.11)}$$

and the output vector is chosen to be

$$y = \begin{bmatrix} V_T & \gamma & \theta \end{bmatrix}^T \qquad \textbf{(3.12)}$$

where ($\gamma = \theta - \alpha$) is the perturbation in the flight path angle. The output vector shown in equation (3.12) is chosen in order to facilitate the establishment of setpoints required to accomplish the longitudinal maneuvers employed in this thesis. This choice of outputs also ensures that $(\hat{C}, \hat{A})$, the discrete time output and plant matrices of the system

modified to accept control increments, is observable for the flight conditions used in this thesis. It should be noted that these perturbed quantities are not necessarily directly measurable in reality; however, in this thesis they are assumed to be measurable and available for feedback.

3.3 Input-Output and State Scaling

When trying to establish the relative importance of output deviations from a setpoint, it is beneficial to employ units or non-dimensionalized quantities that permit an accurate comparison of these deviations. For instance, a 1 ft/sec deviation in the forward velocity of an aircraft is certainly not comparable to a 1 radian deviation in pitch angle. Additionally, given the substantial differences in the absolute position limits of the control surfaces, scaling the control deflections based on these limits or on other more convenient quantities allows better visualization of relative control usage. After the appropriate scale factors are chosen with regard to a particular trimmed flight condition, the input-output and state scalings may be accomplished through elementary coordinate transformation procedures. Consider the state space realization

$$\begin{aligned} \dot{x}_o(t) &= A_o x_o(t) + B_o u_o(t) \\ y_o(t) &= C_o x_o(t) \end{aligned} \quad , \tag{3.13}$$

with original state, input, and output quantities (x_o, u_o, y_o). Their scaled counterparts (x, y, u) are related to the original quantities through the relations

$$\begin{aligned} x_o &= Tx \\ u_o &= Su \\ y_o &= Ry \end{aligned} \tag{3.14}$$

where T, S, and R are transformation matrices of appropriate dimensions. Substituting the transformation equations from (3.14) into equation (3.13) yields the following scaled system

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{3.15}$$

where

$$A = T^{-1}A_oT \qquad B = T^{-1}B_oS \qquad C = R^{-1}C_oT \tag{3.16}$$

In this thesis, the state transformation from radians to degrees was the first step in the scaling process. Next, with the exception of the leading edge and trailing edge flaps, the inputs were scaled based on twice the absolute difference between the trim deflection and the nearest absolute position limit. The factor of two comes from the definition of the symmetric control deflection in equation (3.6). The leading edge and trailing edge flaps are special cases because, although the nearest absolute position limits are upward (negative) deflections, the movement of these control surfaces during the commanded maneuvers is primarily in the downward (positive) direction. Therefore, scaling factors for these two control surfaces were chosen to enhance the clarity of the control signal graphs and then multiplied by a factor of two for consistency with the other aerodynamic

control surfaces. Finally, the outputs were scaled to depict the deviations of the flight path angle and the pitch angle from the trim condition in degrees, and the scaling of the deviation in forward velocity from trim was determined based on an estimation of its relative importance with respect to a deviation in the flight path and pitch angles. A complete listing of all the scaling factors for the flight condition used in this thesis is contained in Table 3.5.

One consequence of input-output scaling that must be taken into account is its effect on the performance index used in the quadratic optimization procedure delineated in Section 2.3.6. Recall the performance index, $J(k)$, is defined by

$$J(k) = \sum_{l=1}^{p} \left\| \hat{C}\hat{x}(k+l) - s(k+l) \right\|_{R_y}^2 + \sum_{l=1}^{q} \left\| \Delta u(k+l-1) \right\|_{R_u}^2 \qquad \textbf{(3.17)}$$

Now, note that each term in the summation of the control increment inputs over the control horizon is of the form

$$\Delta u(k+l)^T R_u \Delta u(k+l) \qquad \textbf{(3.18)}$$

where Δu refers to the scaled control increment inputs and R_u is a weight on control usage. Substituting the control scaling, $\Delta u_o = S\Delta u$, in which Δu_o refers to the unscaled control input increments, into equation (3.18) then yields the modified cost term

$$\Delta u_o(k+l)^T S^{-T} R_u S^{-1} \Delta u_o(k+l) \qquad \textbf{(3.19)}$$

which clearly indicates a new weighting factor, $R = S^{-T} R_u S^{-1}$, is being applied to the unscaled control increment inputs. Thus, the scaling factors distinctly influence the distribution of control power to those control surfaces with the largest scaling factors. To rectify this situation, the desired control power weighting should be established in R, and R_u should then be calculated by

$$R_u = S^T R S \tag{3.20}$$

Unless otherwise indicated, R will always be chosen as a constant, c, times the identity matrix in this thesis, which further simplifies equation (3.20) to

$$R_u = c S^T S \tag{3.21}$$

The scaled and unscaled models may be found in Appendices A1 and A2.

Table 3.5: Scaling Factors

Variables	Flt. Cond. 1
V_T	1/8 ft/sec
γ	1/1 deg
θ	1/1 deg
δ_{TVS}	1/20 deg
δ_{AS}	1/50 deg
δ_{SS}	1/34 deg
δ_{LES}	1/30 deg
δ_{TES}	1/60 deg
δ_T	1/27 deg

3.4 Flight Condition

The flight condition utilized in this research effort represents level flight at an altitude of 15,000 feet at Mach 0.24. It is expected to be a challenging case because the low airspeed and the high angle of attack diminish the effectiveness of the aerodynamic control surfaces and also because the F-18 HARV has an unstable phugoid mode at this trim point. As a consequence of the decreased aerodynamic control authority, thrust vectoring should play an important role in performing the selected maneuvers. Table 3.6 contains the initial aircraft states and control deflections for this trim point, and Table 3.7 lists its open loop poles, damping ratio, and natural frequency.

Table 3.6: Trim Flight Conditions

Parameter	Op. Point 1
Altitude (ft)	15000
Mach Number	0.24
V_T (ft/sec)	238.7
α (deg)	25
q (deg/sec)	0
θ (deg)	25
γ (deg)	0
$\delta_{TVS} = \delta_{TVL} + \delta_{TVR}$	0 + 0
$\delta_{AS} = \delta_{AL} + \delta_{AR}$	0 + 0
$\delta_{SS} = \delta_{SL} + \delta_{SR}$	(-6.40) + (-6.40)
$\delta_{LES} = \delta_{LEL} + \delta_{LER}$	(0.33) + (0.33)
$\delta_{TES} = \delta_{TEL} + \delta_{TER}$	(0.80) + (0.80)
δ_T	100.5

Table 3.7: Open Loop Poles	
Parameter	**Op. Point 1**
Phugoid poles	0.0188 ± 0.1280j
Phugoid damping	N/A
Phugoid nat. freq.	0.13
Short period poles	-0.2481 ± 0.3585j
Short period damping	0.57
Short period nat. freq.	0.44

3.5 Precision Longitudinal Maneuvers

The three longitudinal precision control modes of interest to fighter aircraft employed in this thesis are vertical translation, pitch pointing, and direct lift, of which descriptions are taken from [7]. During a vertical translation maneuver, the pitch angle is held constant while the flight path angle is varied. Pitch pointing is characterized by a change in pitch angle with no variation in flight path angle. Lastly, direct lift is a mode in which the flight path angle and the pitch angle change so that angle of attack is held constant. The setpoints based on the outputs specified in equation (3.12) may be found in Table 3.8.

Table 3.8: Setpoint Specifications	
Vertical Translation	$s^{\infty} = [\,0\;1\;0\,]^T$
Pitch Pointing	$s^{\infty} = [\,0\;0\;1\,]^T$
Direct Lift	$s^{\infty} = [\,0\;1\;1\,]^T$

4.0 Simulation Results

The following sections present the results of simulations performed in order to determine the effectiveness of a state space MPC controller at handling actuator failures. All six available control inputs are used in every simulation because this scenario offers the greatest amount of control redundancy, which is especially important in the case of out-of-trim failure conditions. To simulate these failures, multiple sets of constraints are formulated, and at the time of the simulated failure, the optimization function is updated to operate using the constraint set that defines the failure condition. Additionally, if necessary to improve performance in the post-failure operating environment, the weighting matrices R_y and R_u may be updated in the optimization through the introduction of new S and T terms as shown in equations (2.45) and (2.46).

Failures are typically, but not always, specified by setting the rate limit on a particular control element to a very small, but non-zero value. In this fashion, the failures are made to occur at natural points along the travel of the control surfaces, and artificially extreme or conservative failure cases are avoided. In general, single element failures are specified at the maximum travel of the specific element during a longitudinal maneuver. The time and magnitude of multiple element failures are determined on a case by case basis.

Because of the results in [4], constraints are applied only at the current time, k. Application of constraints across the entire prediction and control horizons often leads to

a more conservative approach to the tracking problem, which is undesirable with regards to a fighter-type aircraft. Additionally, when quadratic programming is used as the method for finding the series of quasi-reference signal inputs, infeasibility of the optimization results if the system is over-constrained. For the purposes of this thesis, feasibility is a precondition imposed on all the simulations presented here. For information regarding the handling of infeasibility, see chapter 6 in [4].

MPC with quadratic programming as the optimization method is very susceptible to problems related to numerical conditioning and constraint application. Overly stringent constraints will prevent the optimization from finding a feasible solution and the system will consequently become unstable. Therefore, when failures are specified, care should be taken to not set the upper and lower rate limits to zero, for example. Instead, rate limits of 10^{-4} to 10^{-8} are sufficient to make the control surface ineffective and generally do not lead to numerical instability. Furthermore, the optimization weightings should not be chosen such that the quadratic programming matrices are numerically ill conditioned. The results of the simulations will be unreliable and instability often occurs.

Many simulations were performed for this thesis in which the attempt was made to decrease the sample time of the system and thus improve overall system tracking performance. In all cases, decreasing the sample rate below 0.5 seconds did not improve performance and very often resulted in worse performance. The reason is primarily that the pole placement algorithm becomes less and less efficient as the sample time is decreased, causing the poles to be further from the origin than is desirable. Hence, FIR

characteristics are diminished, and tracking performance suffers. Another result of the lower sample time is to make the system very susceptible to changes in the optimization weightings. Small changes in the weights often lead to large overshoots and/or non-minimum phase behavior.

All relevant Matlab functions are contained in Appendices B1 through B8. A diagram of the simulink model used to perform the simulations is contained in Appendix C.

4.1 Vertical Translation

As mentioned earlier, vertical translation represents a change in the flight path angle of the aircraft while holding the forward velocity and the pitch angle constant. The added lift production is primarily from the ailerons and the trailing edge flaps, with all the constrained cases favoring the ailerons because of their more responsive rate limit. The results of [4,7] and the simulations in this thesis indicate that thrust vectoring is used primarily to maintain the proper pitch angle and minimize deviations of the forward velocity from the trim condition. In all vertical translation cases, $R_y = \text{diag}(50,20,10^4)$, $R_u = (10^{-6})S^TS$, and the sample rate is 0.5 seconds. The small control weightings increase the overall speed of the system response and often cause the control surfaces to reach their rate and/or position limits during the aggressive tracking behavior that results. The commanded input is a unit pulse with a duration of five seconds.

Figures 4.1 through 4.4 represent a case in which the optimization is performed without imposing any saturation or rate limits on the control surfaces. The aircraft

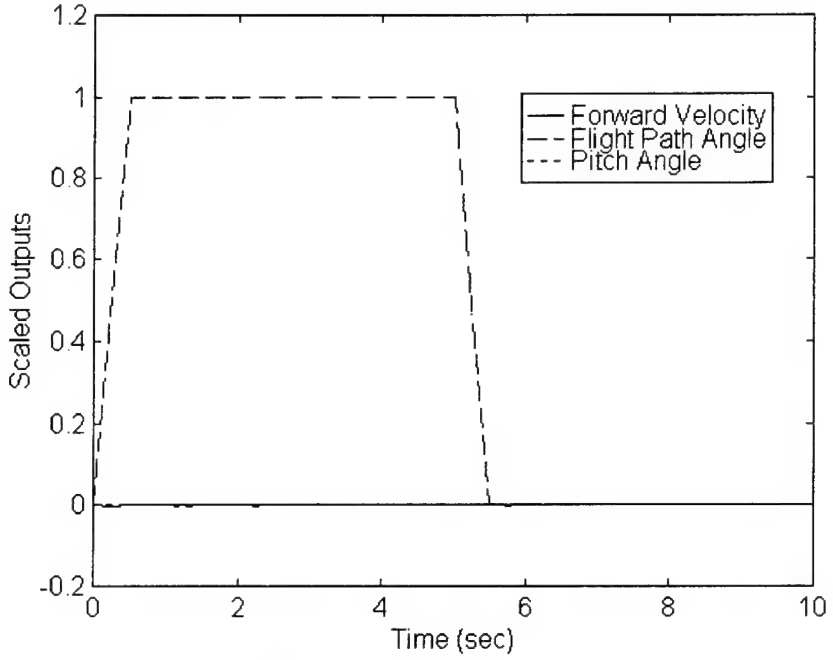


Figure 4.1 VT Output Response (Unconstrained)

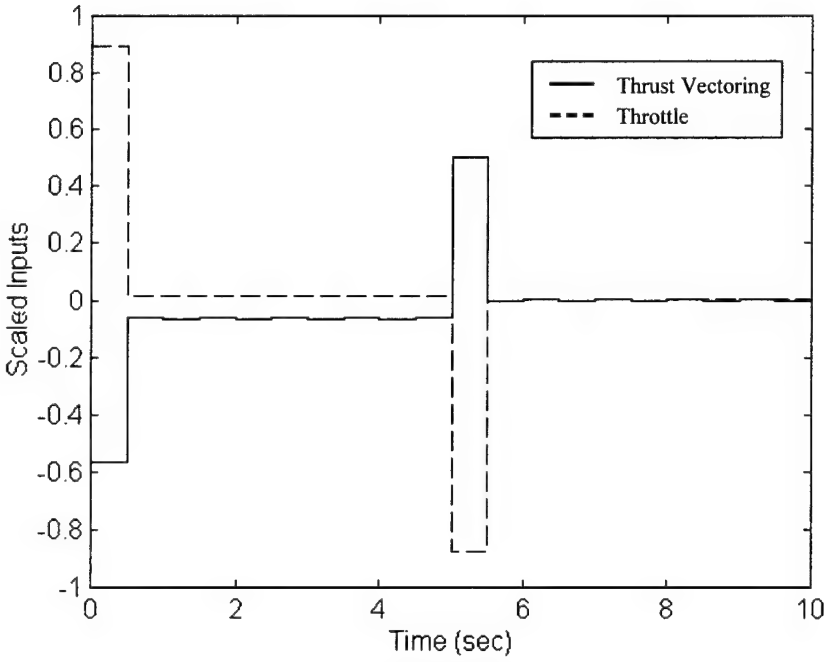


Figure 4.2 VT Control Deflections (Unconstrained)

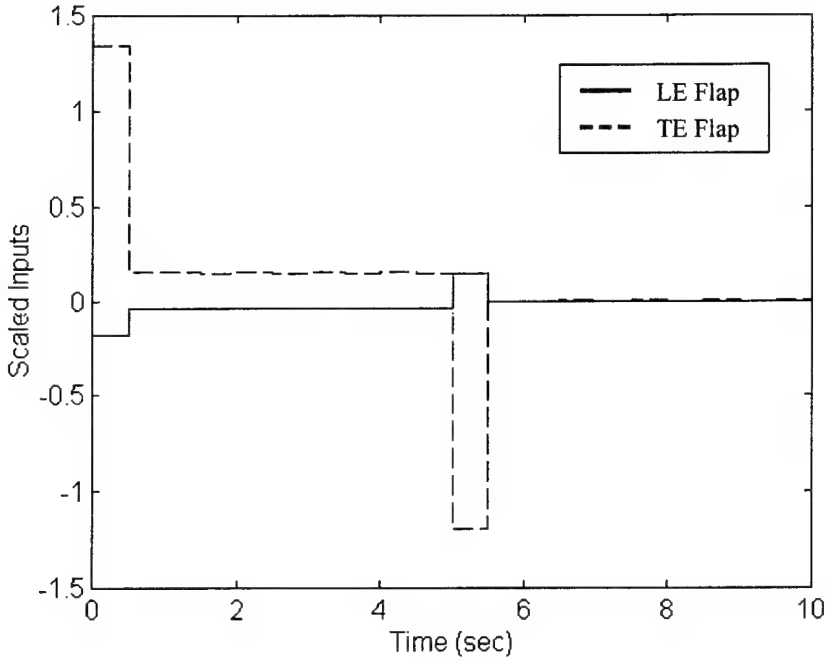


Figure 4.3 VT Control Deflections (Unconstrained)

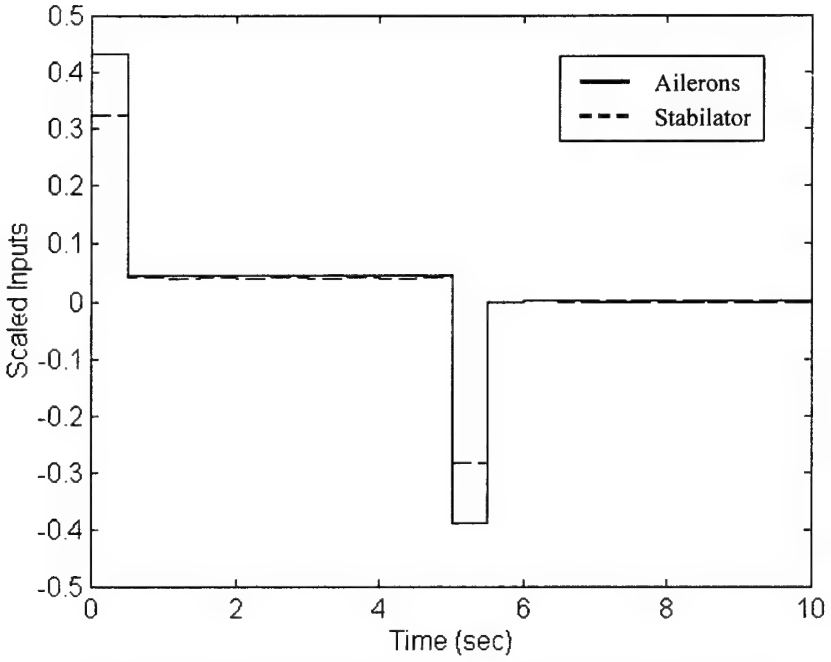


Figure 4.4 VT Control Deflections (Unconstrained)

reaches its setpoint within 0.5 seconds, but not without violating rate limits on both the throttle and the trailing edge flaps and the upper saturation limit on the trailing edge flaps. The application of the nominal constraint set, as shown in Figures 4.5 through 4.8, produces a large increase in overall control usage. Because the flaps are now heavily constrained by their rate limit, the ailerons are forced to their maximum downward position limit in order to produce the lift required to accomplish the maneuver. Furthermore, during the initial maneuver to reach the setpoint, rate limits are encountered in both the trailing edge flaps and the throttle, and the thrust vectoring vane attains its upward position limit in order to counteract the effects of the negative pitching moment produced by the downward deflections of the ailerons and flaps. Similarly, when the aircraft is commanded to return to its original equilibrium condition at time t = 5 seconds, the ailerons are driven to their maximum upward position limits in order to dissipate lift, while the thrust vectoring vanes attain their maximum downward limit to counteract the positive pitching moment generated by the flaps and ailerons and to prevent large deviations in forward velocity.

The next three cases involve failures of single control surfaces. In Figures 4.9 through 4.12, the stabilator is frozen at time t = 0.5 seconds. The flaps respond by returning to a position near their original trim point to diminish the effects of the extra lift caused by the out-of-trim failure of the ailerons. Additionally, the throttle is retarded to help prevent the flight path angle from exceeding its setpoint. The stabilator and the thrust vectoring vane are pushed almost to their upper and lower limits, respectively,

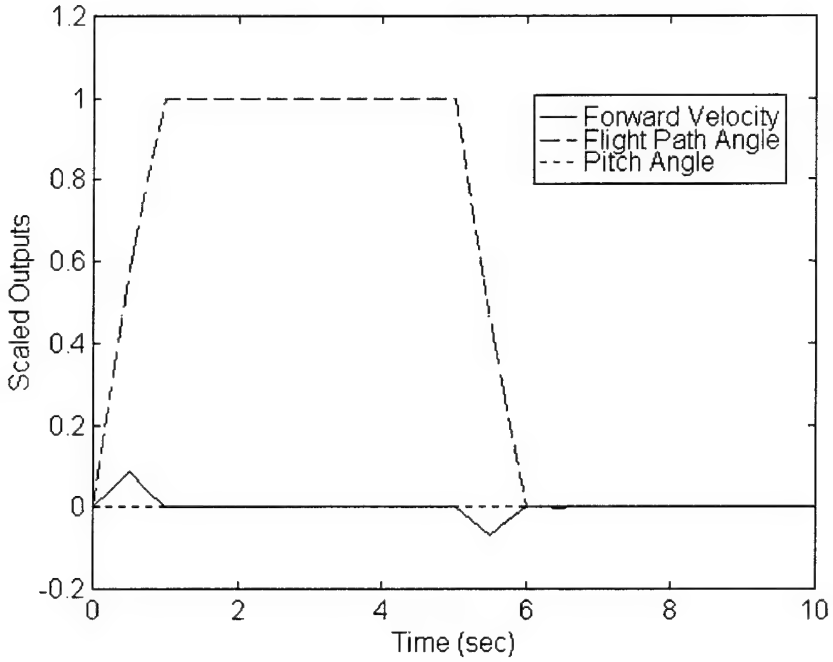


Figure 4.5 VT Output Response (Nominal Constraints)

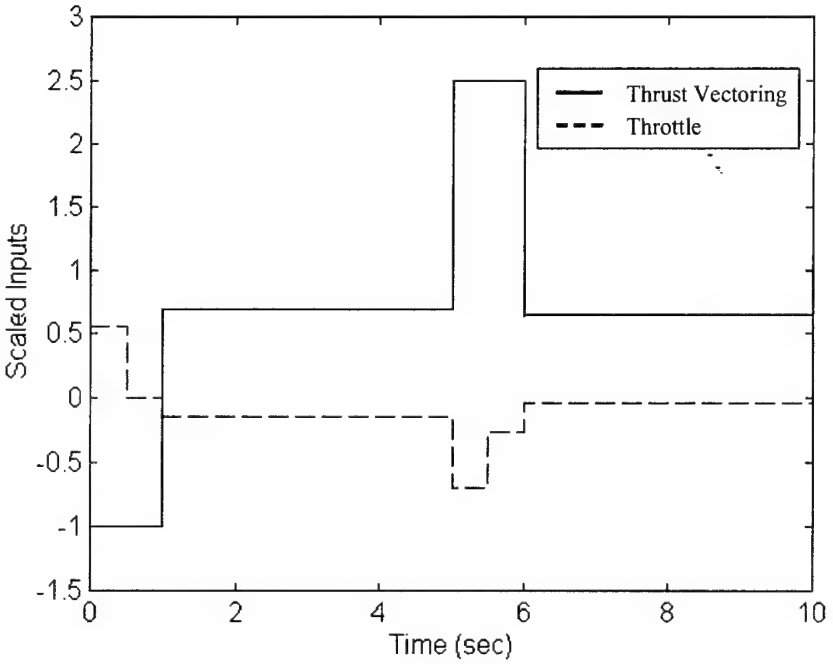


Figure 4.6 VT Control Deflections (Nominal Constraints)

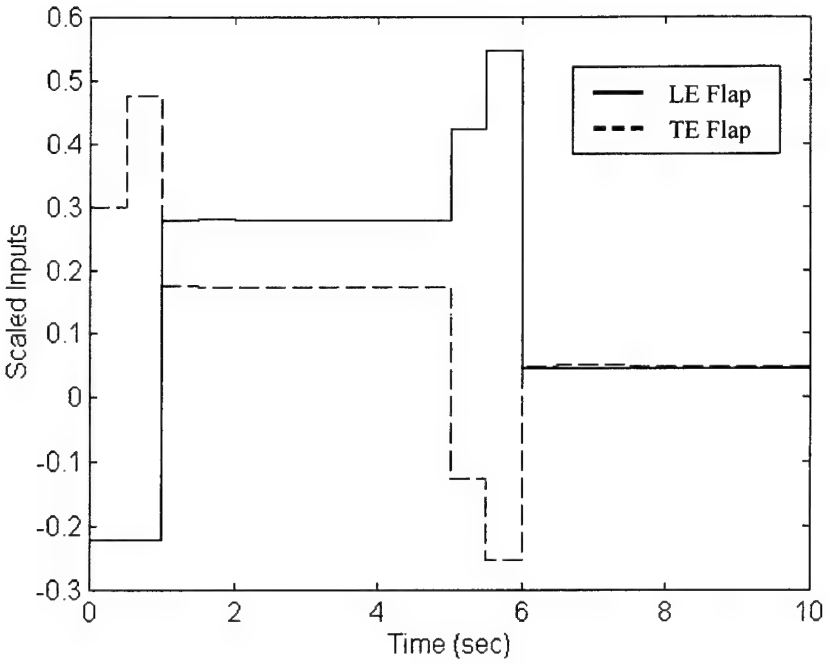


Figure 4.7 VT Control Deflections (Nominal Constraints)

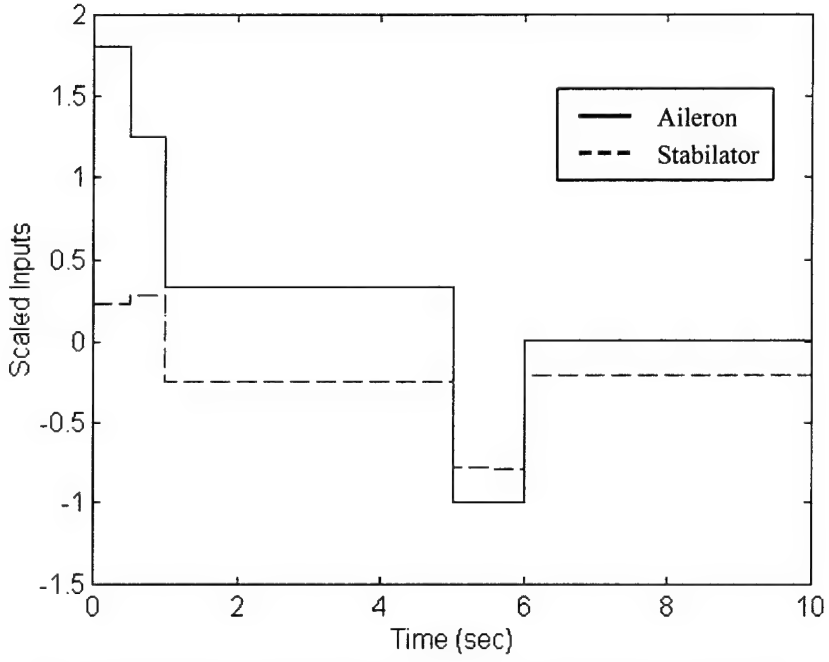


Figure 4.8 VT Control Deflections (Nominal Constraints)

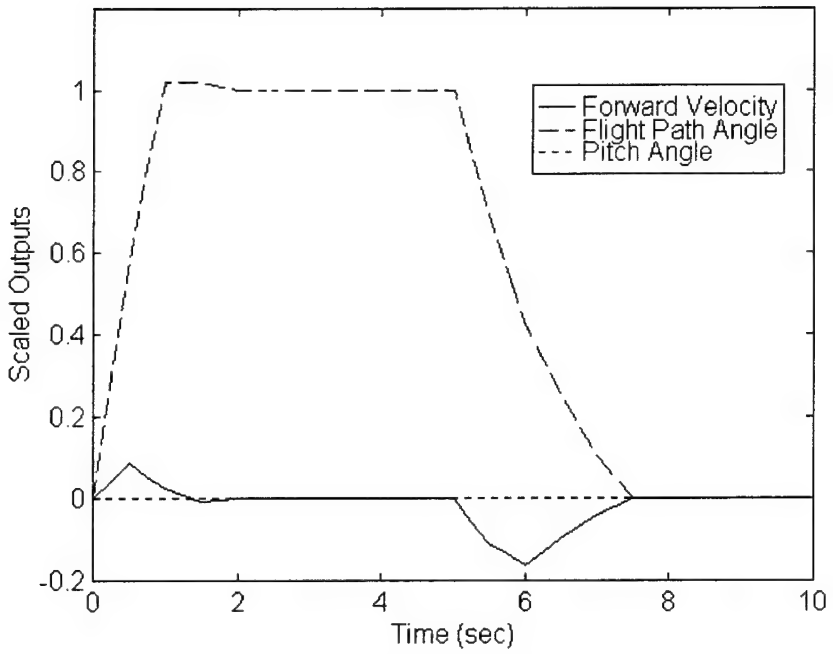


Figure 4.9 VT Output Response (Failure: AS at t=0.5)

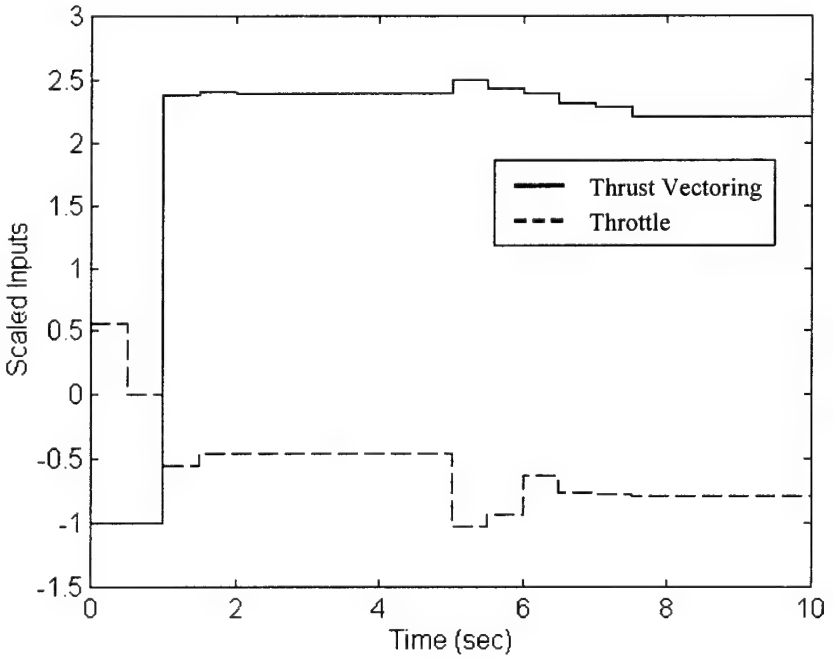


Figure 4.10 Control Deflections (Failure: AS at t=0.5)

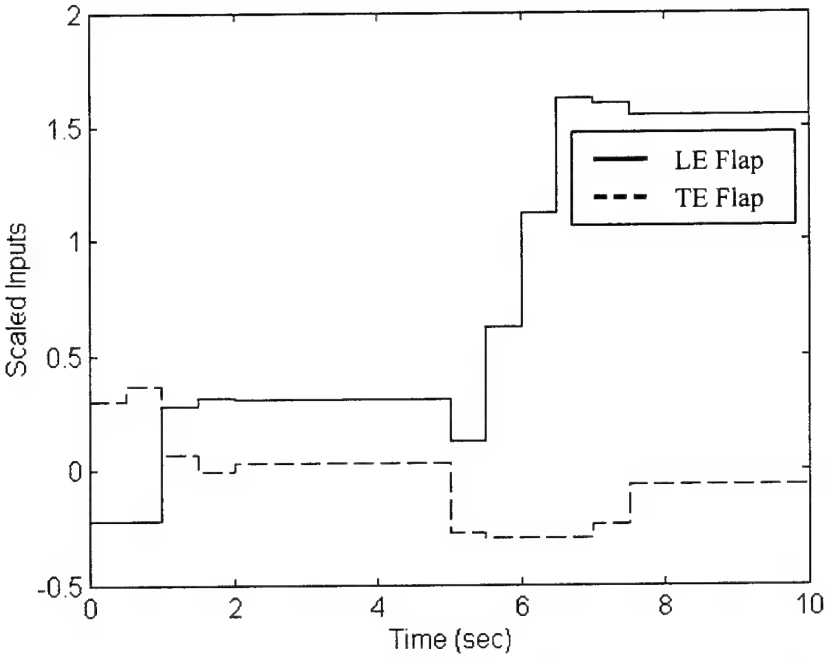


Figure 4.11 VT Control Deflections (Failure: AS at t=0.5)

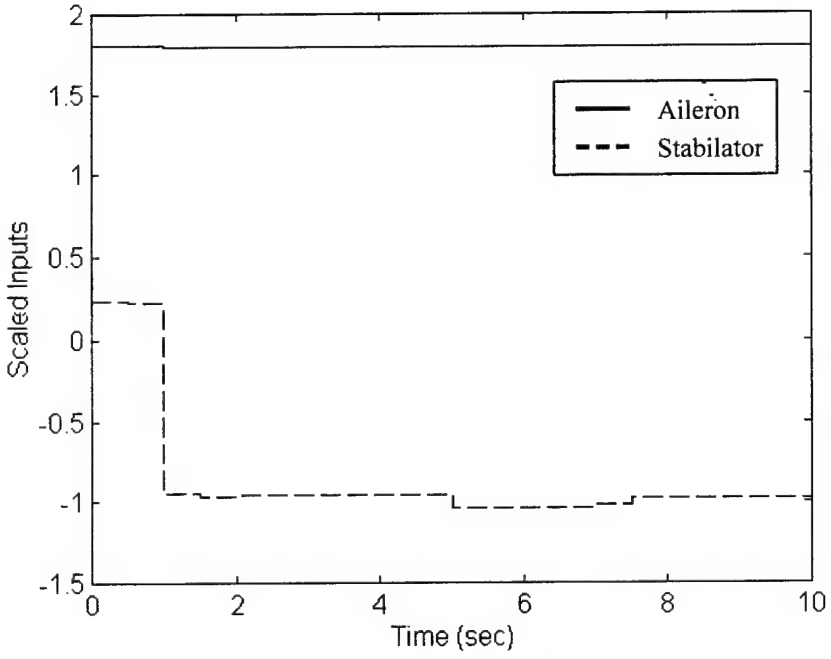


Figure 4.12 VT Control Deflections (Failure: AS at t=0.5)

subsequent to the aileron failure. Most likely, the stabilator is attempting to counteract the effects of the negative pitching moment created by the downward deflection of the ailerons while the thrust vectoring vanes attempt to minimize deviations of the forward velocity from trim. This failure has little effect on the initial setpoint tracking other than a slight overshoot that does not occur in the nominal case. However, the return to the original trim condition takes almost 1.5 seconds longer due to the fact that the flaps must work against their rate limit to dissipate the excess lift.

Figures 4.13 through 4.16 present a failure scenario in which the stabilator fails at time t = 1.0 second. The relatively small deflection at which the stabilator freezes is easily counteracted by the thrust vectoring vane system, and the only noticeable effects on the output response are a slightly longer time required to return to the original trim condition and a greater deviation from the trim forward velocity than in the nominal case.

The last single failure case presented here is a failure of the trailing edge flaps at time t = 1.0 second: Figures 4.17 through 4.20. Of those simulations included in this thesis, this scenario presents the most difficult case for the MPC controller in terms of single control element malfunctions. In response to the update of the controller with the new constraint set, the ailerons are immediately deflected upward in order to shed lift and prevent the aircraft from overshooting its setpoint. Simultaneously, the stabilator deflects upward to counteract the negative pitching moment produced by the flap deflection, and the thrust vectoring vane closely approaches its lower limit to minimize any airspeed deviation. From the output response, we see that there is a slight overshoot, but the

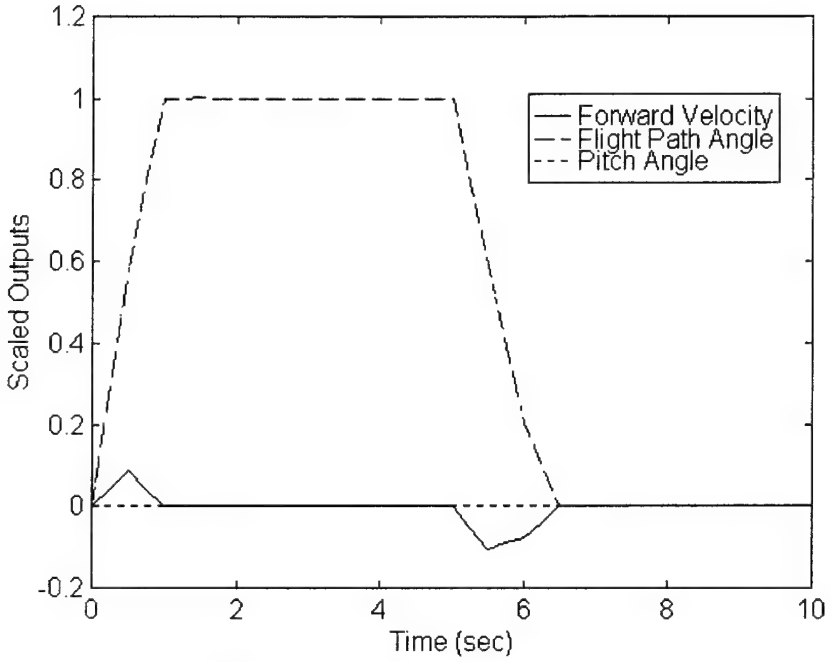


Figure 4.13 VT Output Response (Failure: SS at t=1.0)

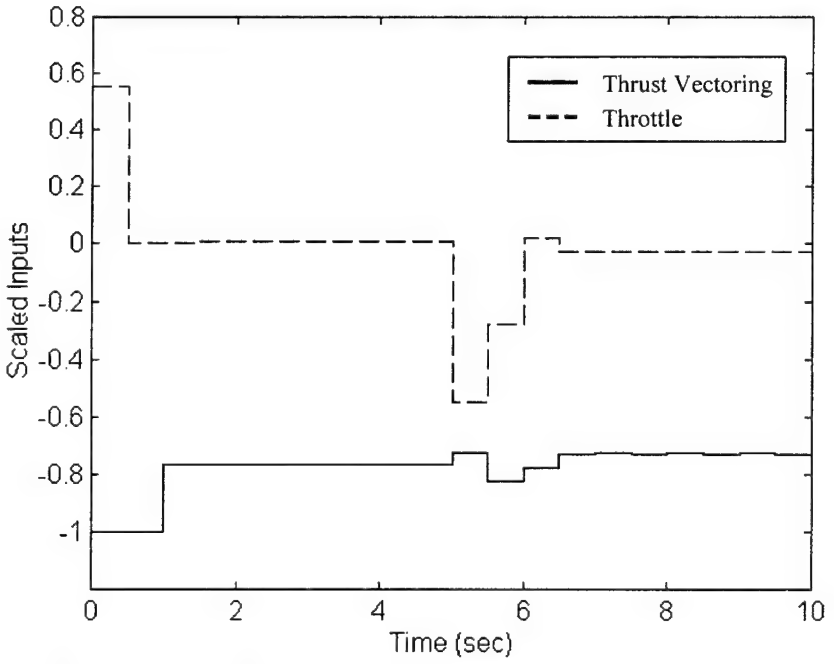


Figure 4.14 VT Control Deflections (Failure: SS at t=1.0)

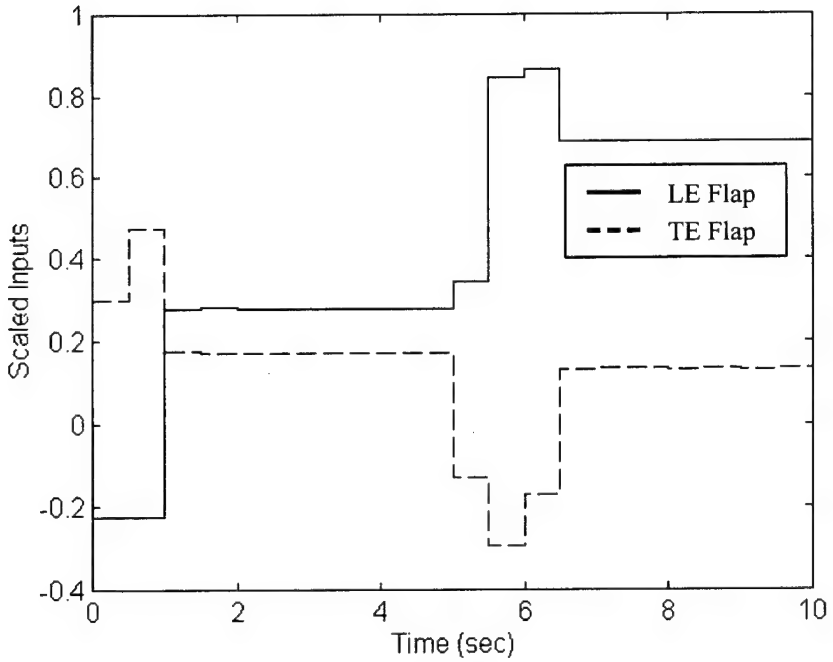


Figure 4.15 VT Control Deflections (Failure: SS at t=1.0)

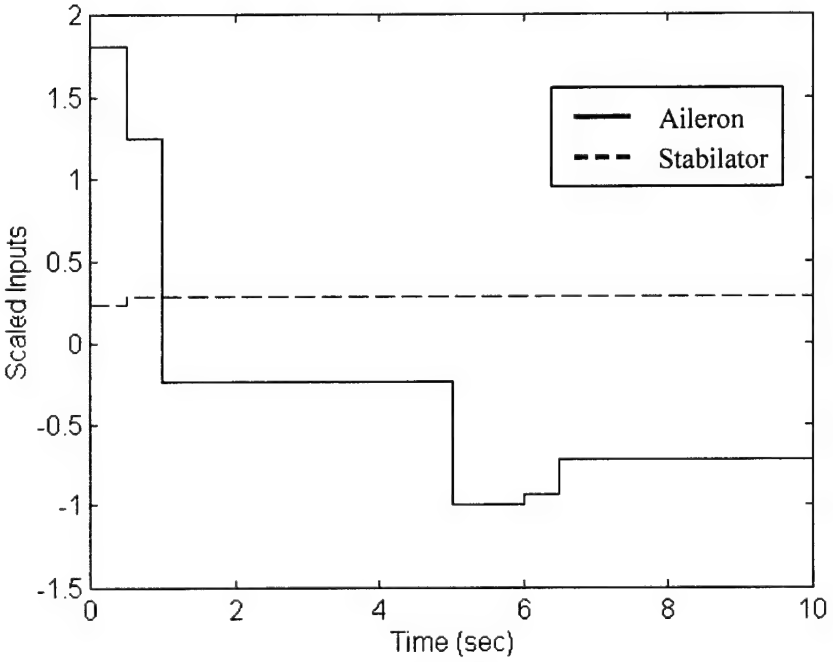


Figure 4.16 VT Control Deflections (Failure: SS at t=1.0)

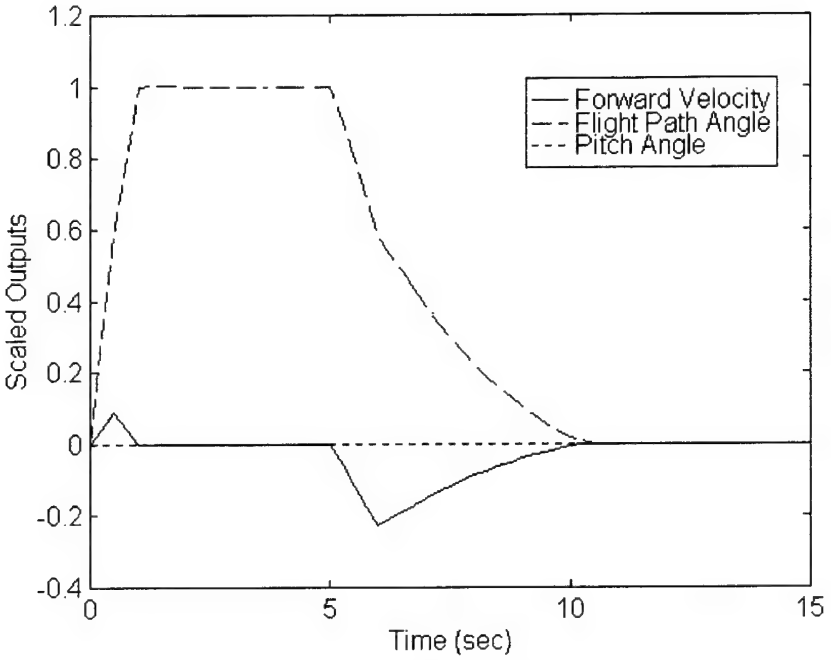


Figure 4.17 VT Output Response (Failure: TES at t=1.0)

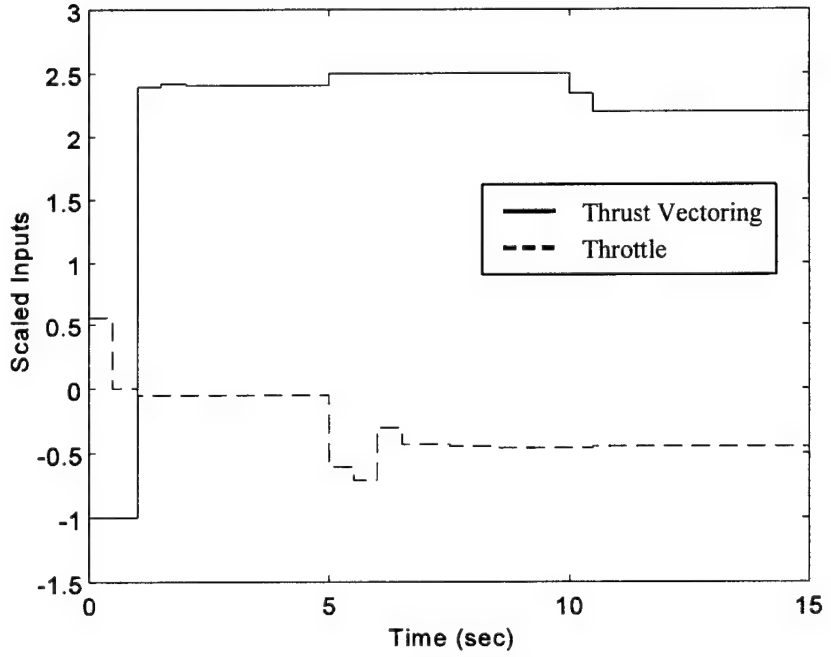


Figure 4.18 VT Control Deflections (Failure: TES at t=1.0)

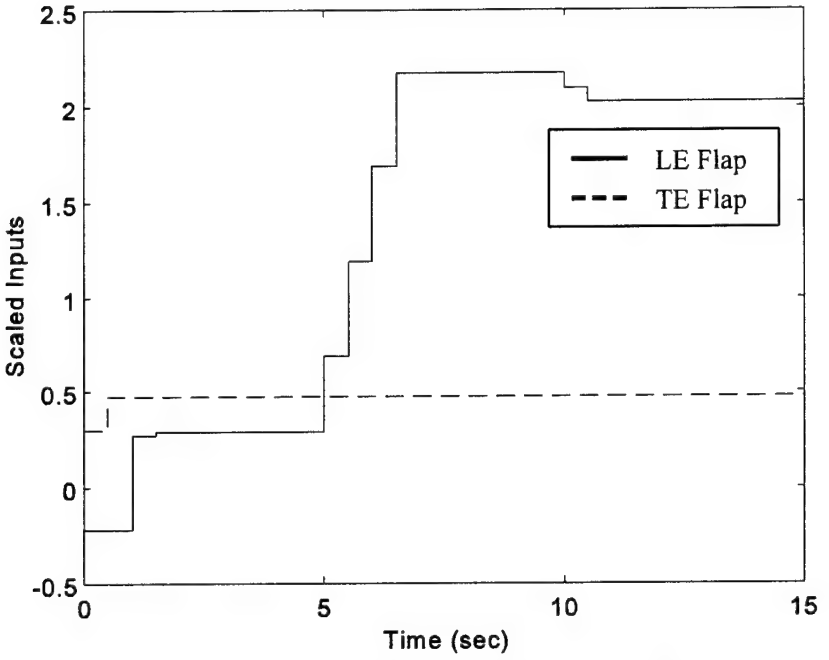


Figure 4.19 VT Control Deflections (Failure: TES at t=1.0)

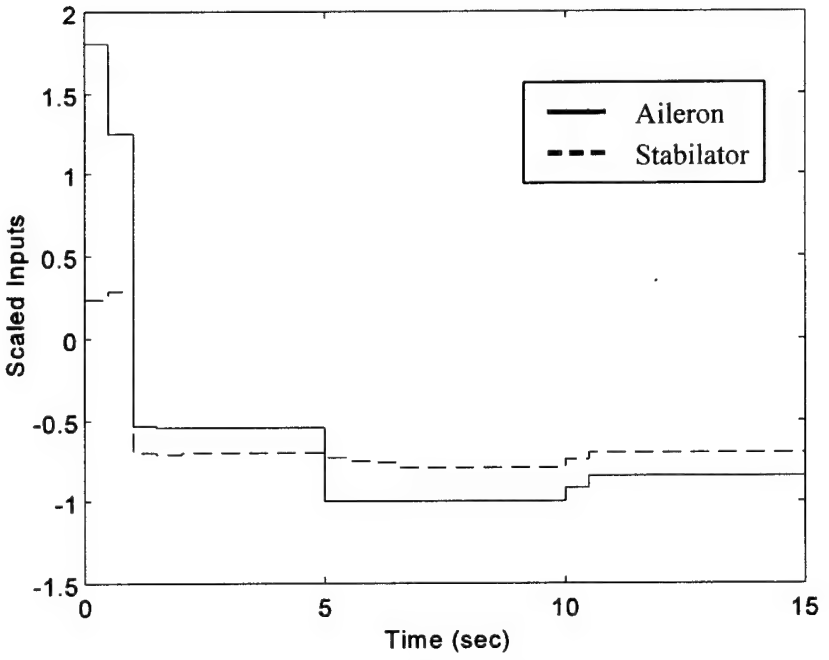


Figure 4.20 VT Control Deflections (Failure: TES at t=1.0)

airspeed deviation from trim is not significantly worse than that of the nominal case. The trouble arises when it is time to return to the original trim condition. The ailerons, already near their maximum upward deflection, are pushed completely to saturation in order to dissipate enough lift to return the flight path angle to horizontal. The throttle is also retarded beyond its position in the nominally constrained case because less power is required to maintain level flight with the flaps deflected.

Now that we have shown the MPC controller is capable of handling single control element malfunctions with minimal performance loss, it is time to explore multiple failure scenarios. The first of these is shown in Figures 4.21 through 4.24. In this instance, the stabilator is assumed to malfunction at time t = 0 seconds and is immediately followed by an out-of-trim failure of the trailing edge flaps at t = 0.5 seconds. An interesting result of a failure of the stabilator at the trim condition prior to a vertical translation maneuver is an implied constraint on the movement of the thrust vectoring vanes. Considering the large weight placed on deviations of the pitch angle from trim in these simulations, the controller cannot move the thrust vectoring vanes to any great degree without significantly altering the pitch angle. In a similar simulation not included here, a failure of the thrust vectoring system prior to the maneuver produces an implied constraint on stabilator movement as well. From the output response, we see that the primary effects of this failure are similar to those of the case where only the trailing edge flaps fail. This scenario does, however, have a longer rise time and a slightly larger deviation in forward velocity from trim during the first one second of the simulation.

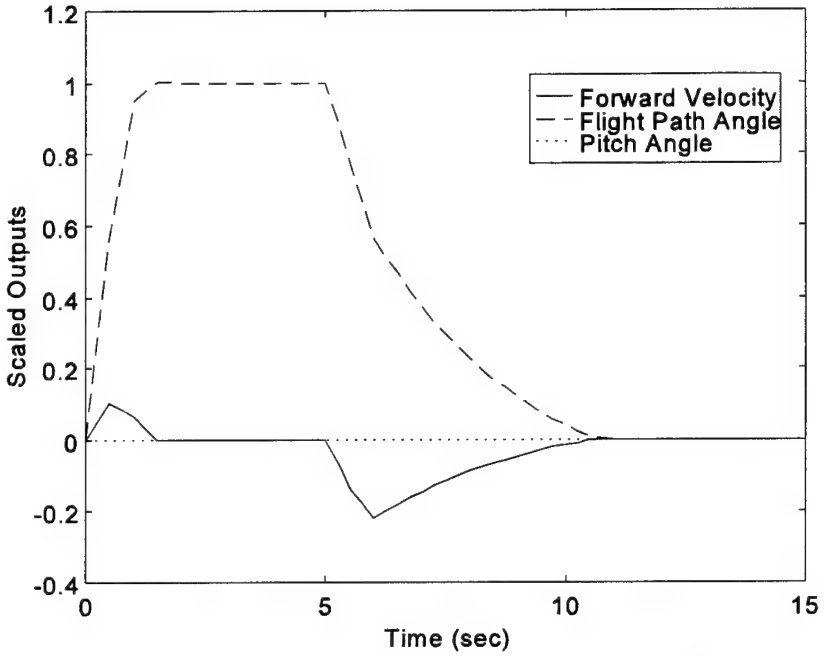


Figure 4.21 VT Output Resp. (Failures: SS at t=0; TES at t=0.5)

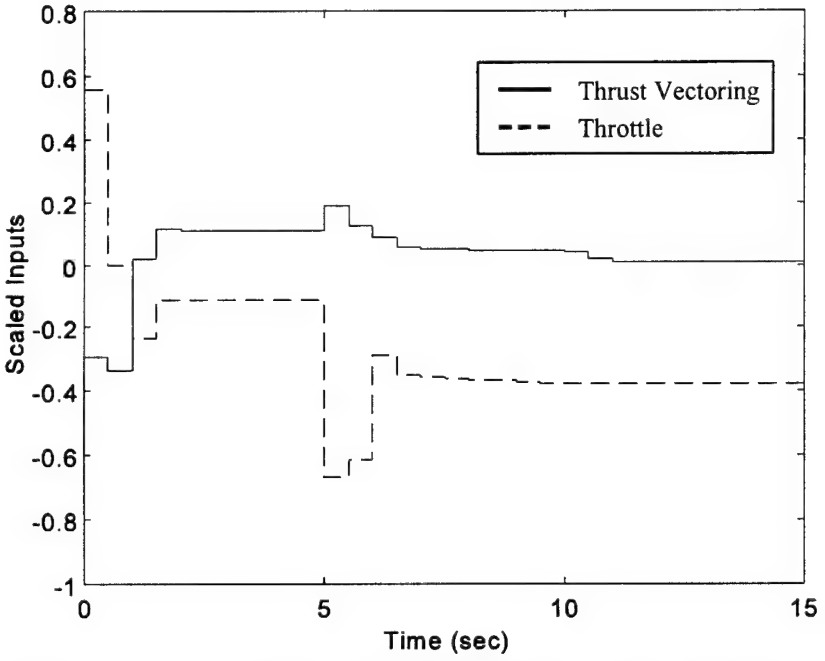


Figure 4.22 VT Control Defl. (Failures: SS at t=0; TES at t=0.5)

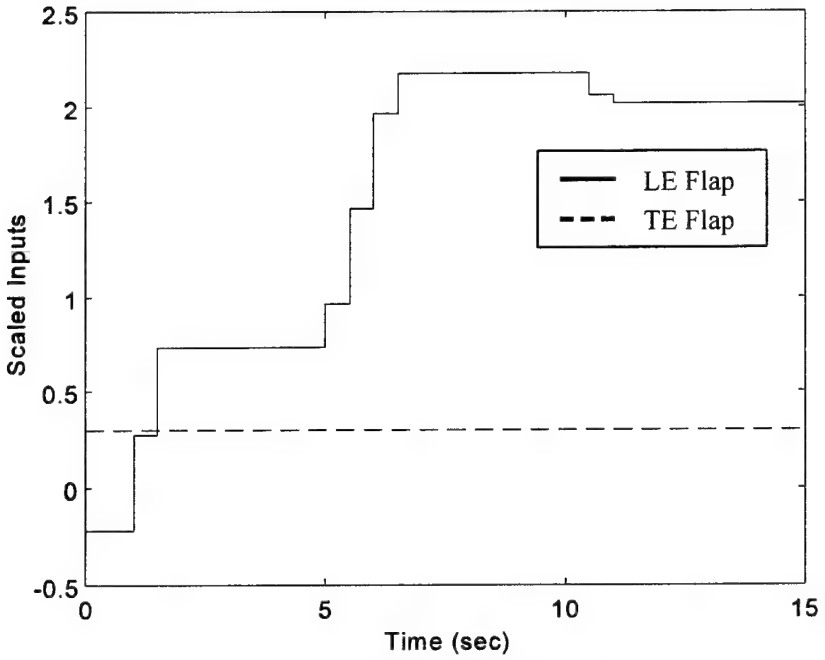


Figure 4.23 VT Control Defl. (Failures: SS at t=0; TES at t=0.5)

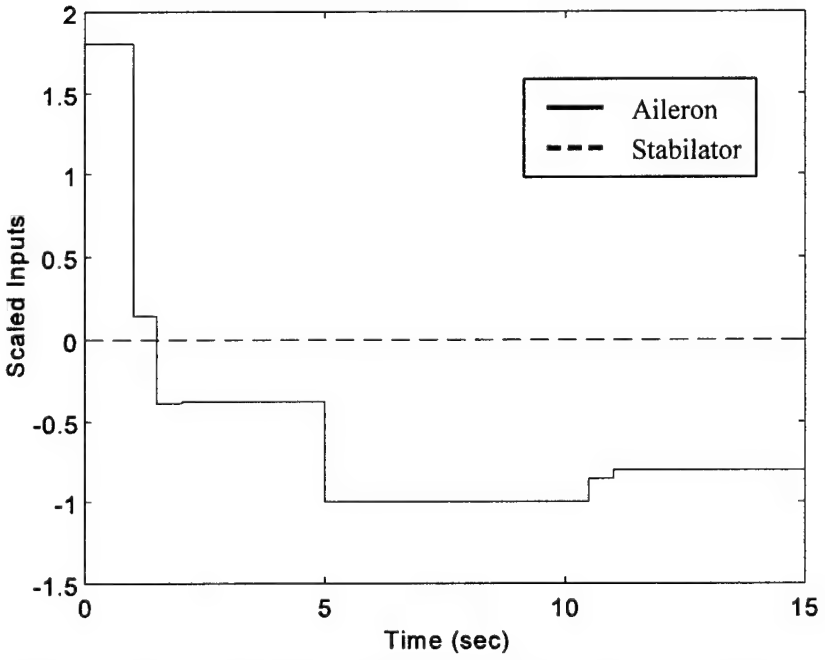


Figure 4.24 VT Control Defl. (Failures: SS at t=0; TES at t=0.5)

The next dual failure condition, shown in Figures 4.25 through 4.28, presents the interesting case of a complete loss of primary pitch control at the trim position. At time t = 0 seconds, the thrust vectoring vanes and the stabilator are frozen and the aircraft is commanded to perform vertical translation. Because vertical translation does not involve a pitch change, the aircraft is still capable of completing the maneuver and returning to its original equilibrium point. From Figures 4.27 and 4.28, we observe that, compared with the nominal case, the maximum aileron deflection is reduced whereas the maximum flap deflection is increased. Presumably this change in lift production distribution occurs in order to diminish the negative pitching moment produced by downward deflections of the flaps and ailerons. The upward deflection of the leading edge flaps appears to counteract the negative pitching moment that is produced by the trailing edge flaps and the ailerons.

The scenario presented in Figures 4.29 through 4.32 is similar to the previous case except that in addition to a complete primary pitch control failure, the trailing edge flaps are frozen at time t = 0.5 seconds. From the output response in Figure 4.29, it is apparent that this series of failures presents a serious hindrance to nominal performance. More importantly, however, the aircraft remains stable and the controller is eventually able to return the aircraft to level flight. Control usage is similar to that of the preceding failure scenario, but a lower throttle setting and a saturated upward aileron deflection are necessary to return the flight path angle to zero.

The last three failure case is displayed in Figures 4.33 through 4.36: the stabilator fails at time t = 0; the leading edge flaps fail at time t = 0.5; the ailerons fail at t = 1.0.

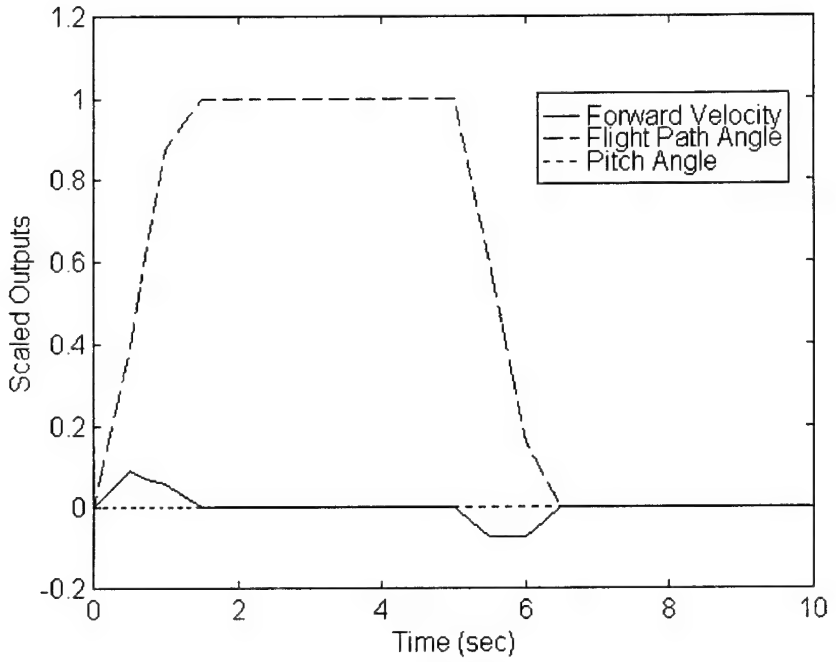


Figure 4.25 VT Output Resp. (Failures: SS and TVS at t=0)

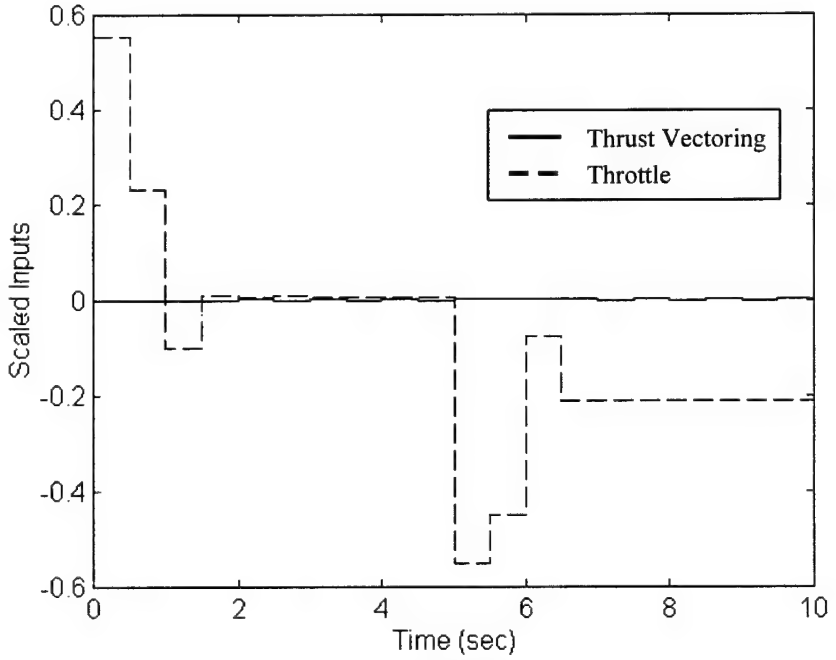


Figure 4.26 VT Control Defl. (Failures: SS and TVS at t=0)

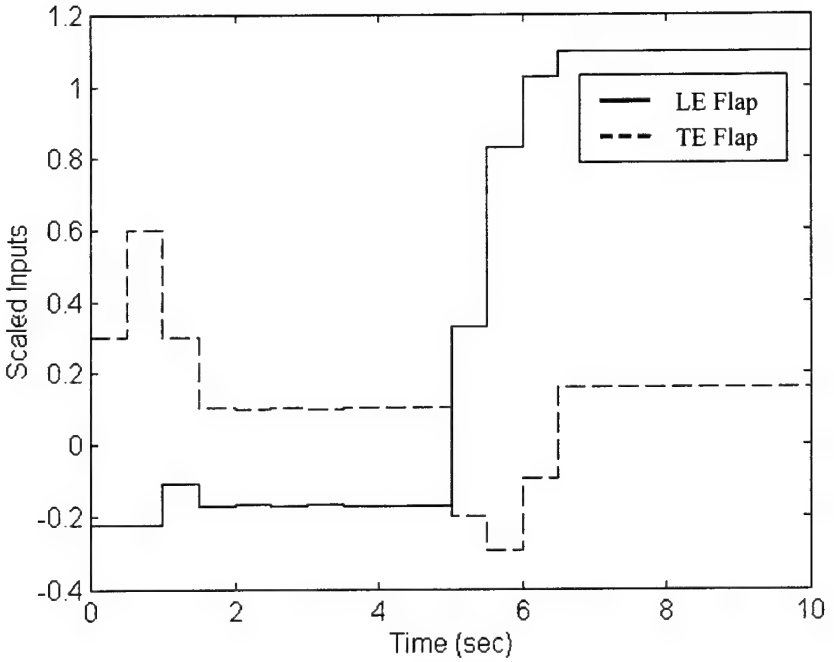


Figure 4.27 VT Control Defl. (Failures: SS and TVS at t=0)

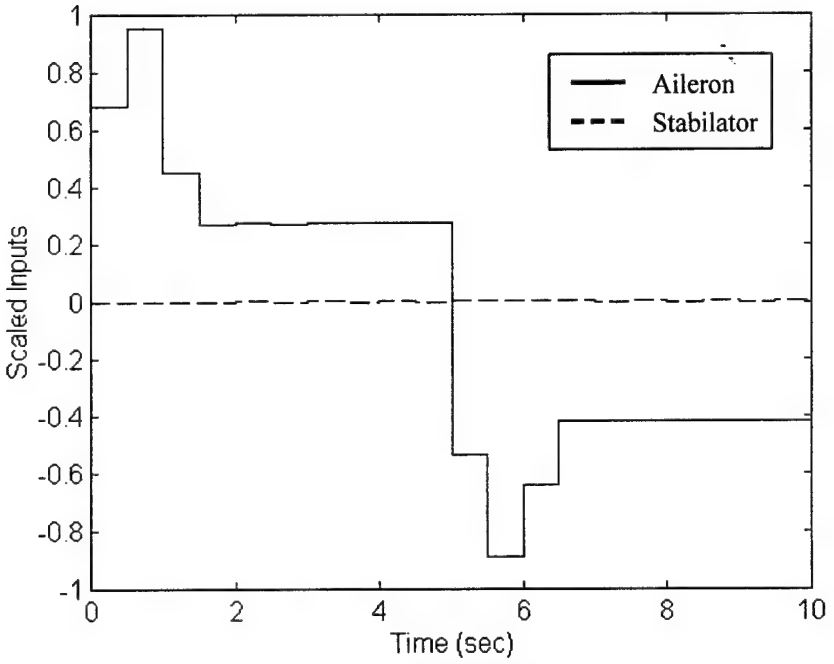


Figure 4.28 VT Control Defl. (Failures: SS and TVS at t=0)

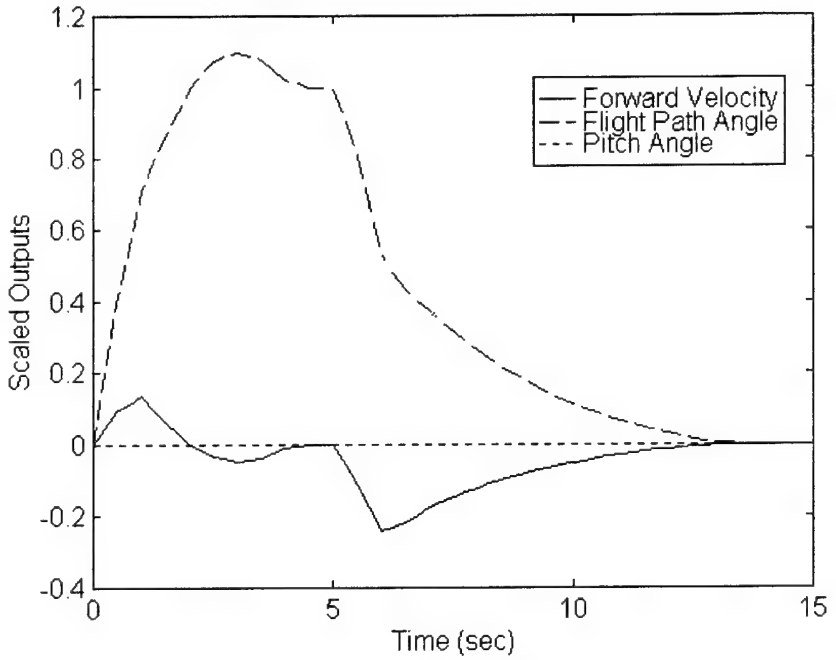


Figure 4.29 Outputs (Failures: SS and TVS at t=0; TES at t=0.5)

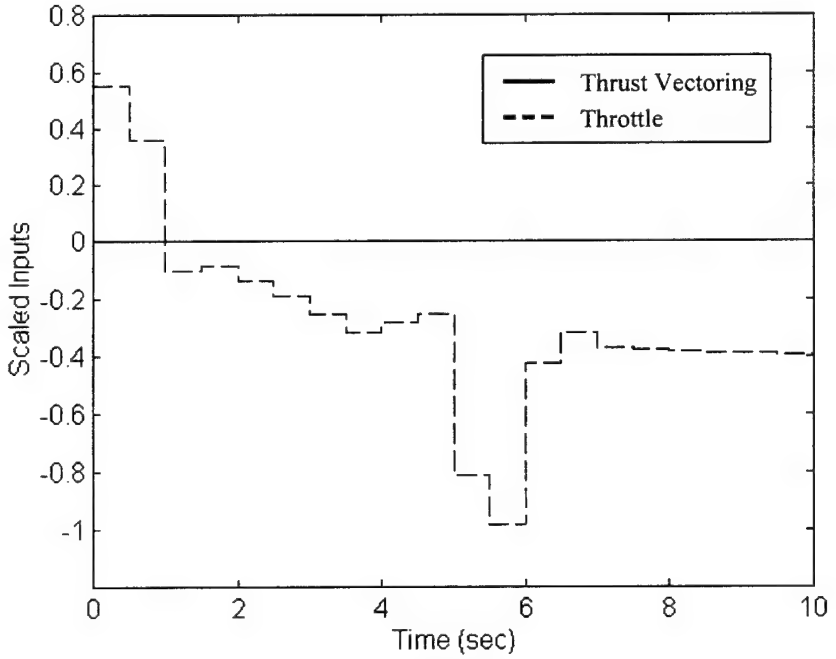


Figure 4.30 Controls (SS and TVS at t=0; TES at t=0.5)

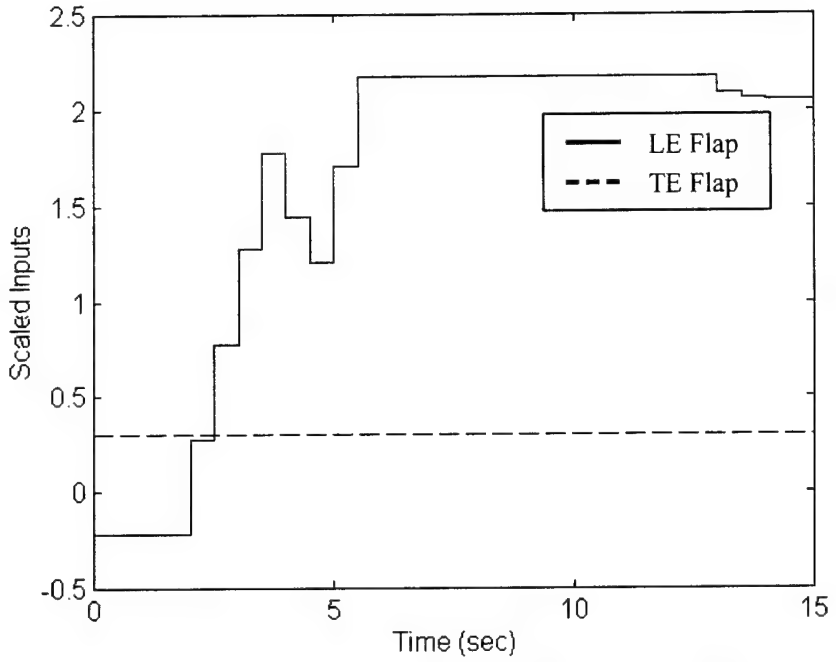


Figure 4.31 Controls (Failures: SS and TVS at t=0; TES at t=0.5)

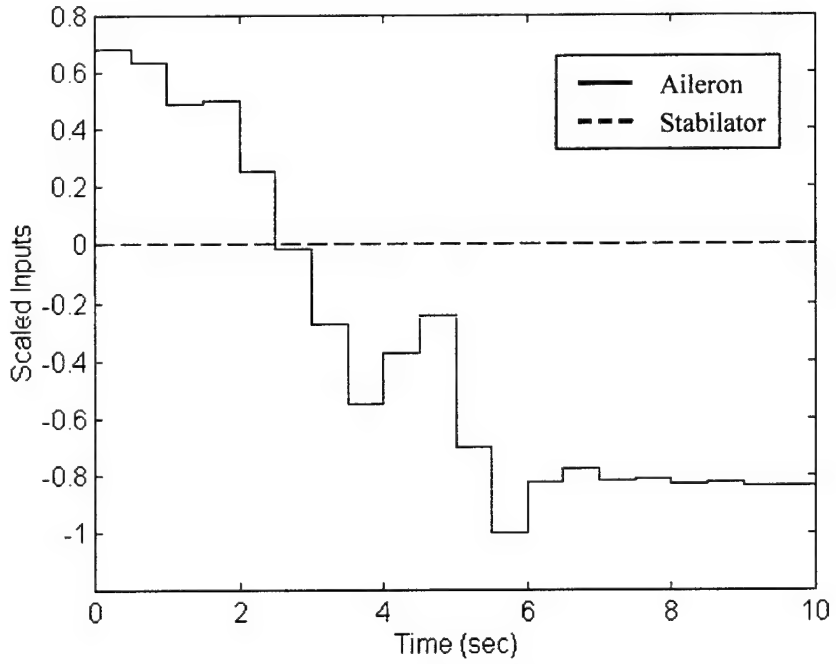


Figure 4.32 Controls (Failures: SS and TVS at t=0; TES at t=0.5)

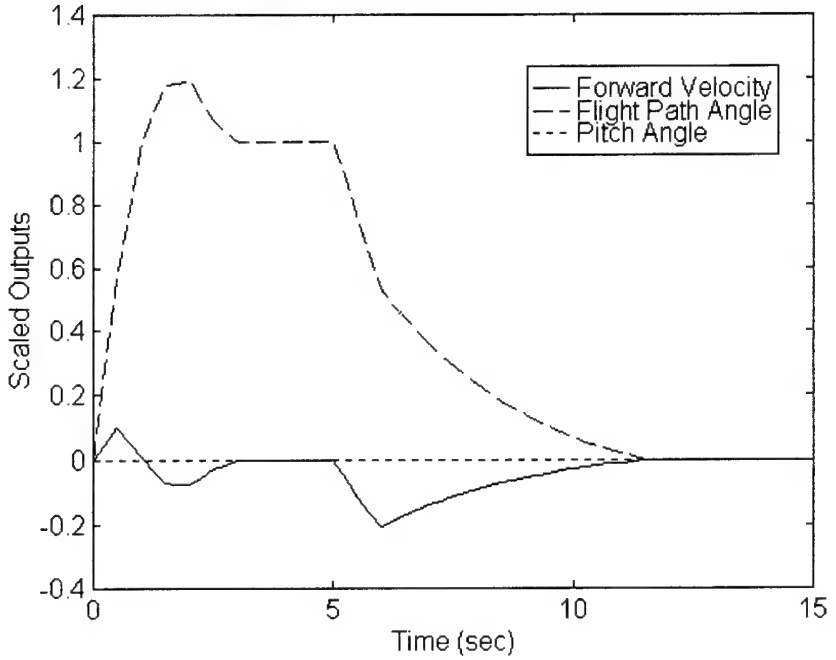


Figure 4.33 Outputs (Failures: SS at t=0; LES at t=0.5; AS at t=1.0)

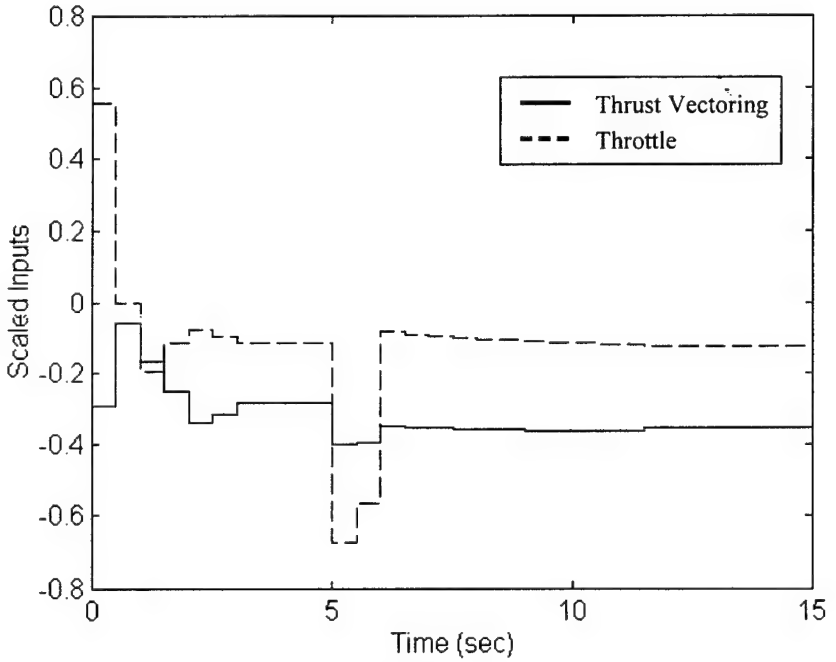


Figure 4.34 Controls (Failures: SS at t=0; LES at t=0.5; AS at t=1.0)

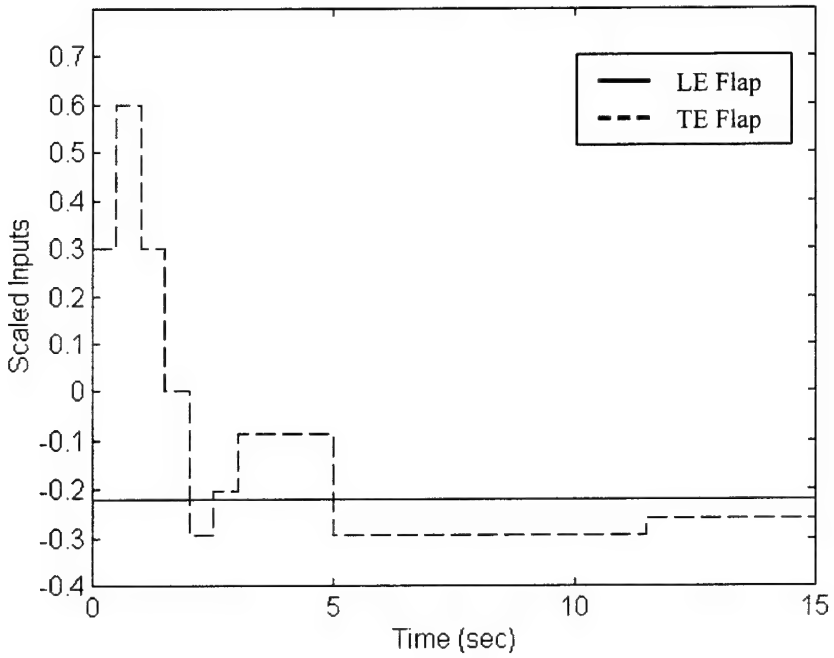


Figure 4.35 Controls (Failures: SS at t=0; LES at t=0.5; AS at t=1.0)

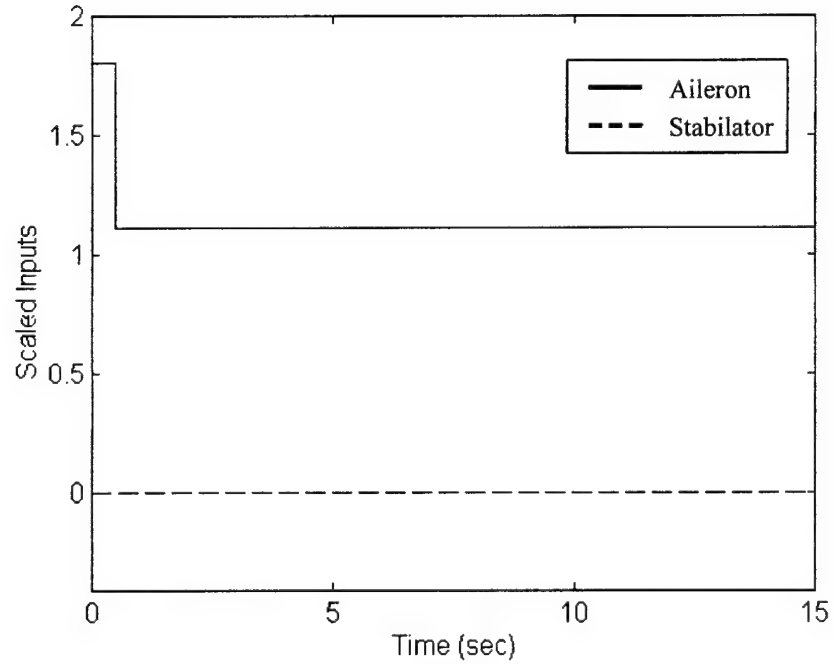


Figure 4.36 Controls (Failures: SS at t=0; LES at t=0.5; AS at t=1.0)

Studying the control inputs in Figures 4.34 through 4.36, we once again see the implied limitation on the deflections of the thrust vectoring vanes because the stabilator is frozen in its trim position and also because of the high weighting term on deviations in pitch angle. However, because of the negative pitching moment produced by the failure of the ailerons in an out-of-trim position, the thrust vectoring vanes are canted upward throughout the simulation. This movement limitation leads to significantly increased deviations in forward velocity from the nominally constrained case and contributes to the overshoot shown in the output response in Figure 4.33. Additionally, the failed ailerons are responsible for the slow return to the original equilibrium condition.

Except for the single control element failures, most of the failure scenarios presented in this thesis are highly unlikely unless the aircraft suffers some form of external damage. However, the benefit from exploring these unlikely cases is derived from the fact that a controller capable of handling such severe failures should be readily capable of handling minor ones. Still, it is worthwhile to explore some instances in which the MPC controller is either unable to return the aircraft to its original trim point or is unable to prevent the aircraft from departing from controlled flight. The first of these cases is shown in Figures 4.37 through 4.40 and represents a failure of the ailerons at $t = 0.5$ seconds and a malfunction of the trailing edge flaps at $t = 1.0$ second. Noting that these two pairs of control surfaces are those primarily responsible for generating the extra lift required for the vertical translation maneuver, we should expect that both tracking the setpoint and returning to equilibrium should be nearly impossible, and this is the case.

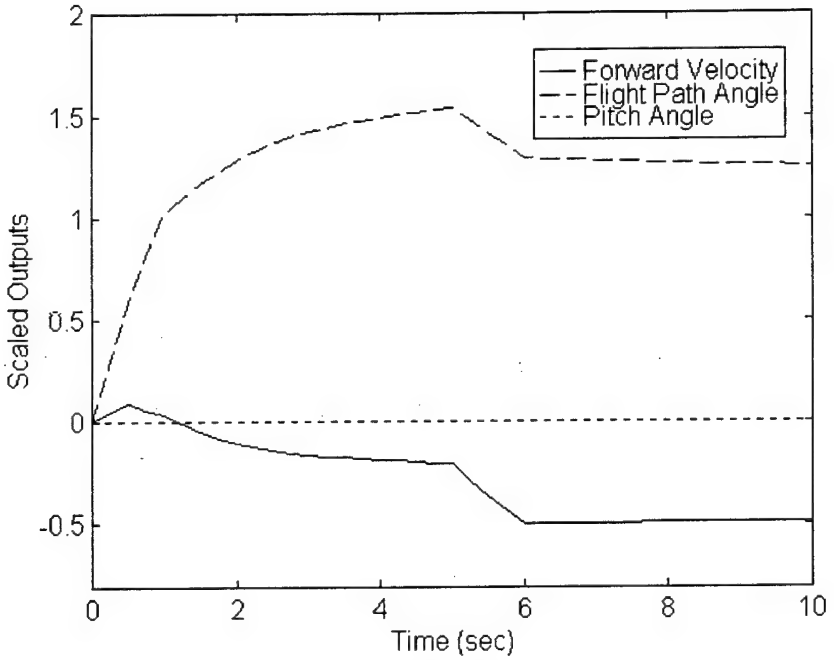


Figure 4.37 VT Output Resp. (Failures: AS at t=0.5; TES at t=1.0)

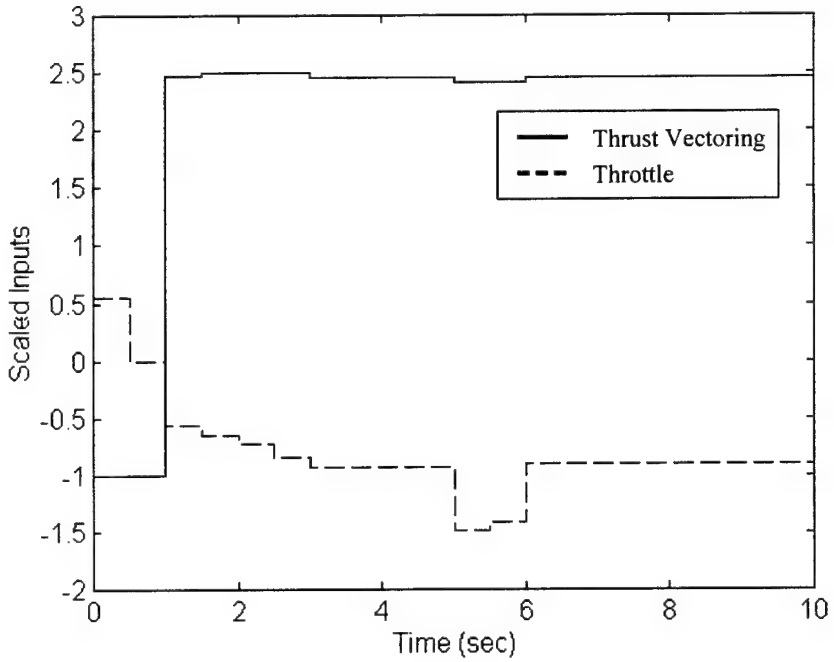


Figure 4.38 VT Control Defl. (Failures: AS at t=0.5; TES at t=1.0)

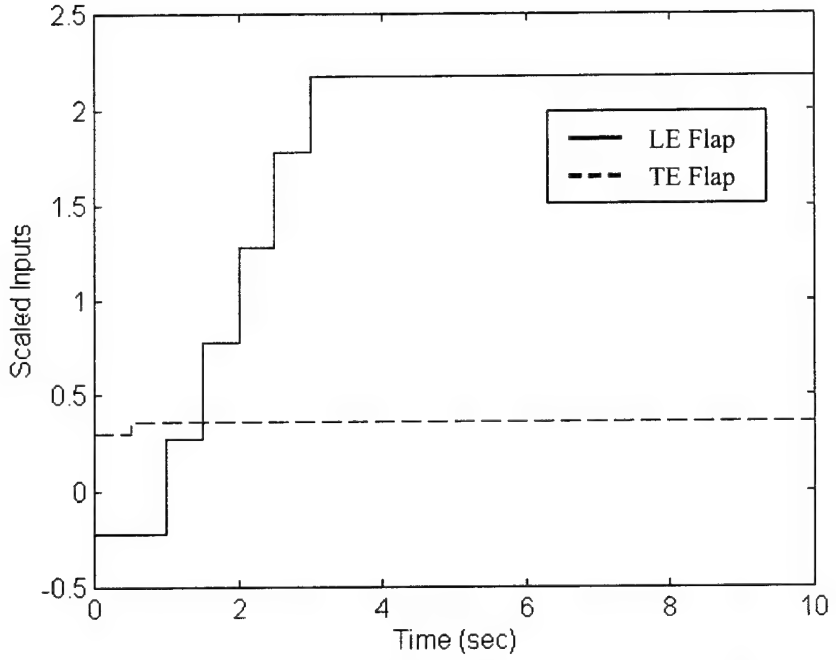


Figure 4.39 VT Control Defl. (Failures: AS at t=0.5; TES at t=1.0)

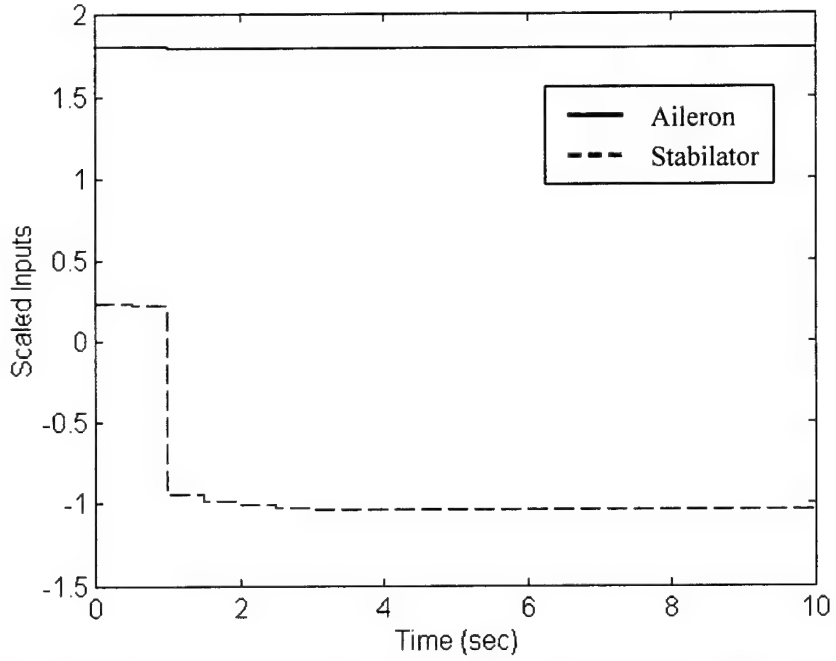


Figure 4.40 VT Control Defl. (Failures: AS at t=0.5; RES at t=1.0)

Subsequent to the trailing edge flap failure at time t = 1.0 second, the stabilator, the thrust vectoring vanes, and the leading edge flaps all reach either their positive or negative saturation limits. As seen in the output response in Figure 4.37, even these extreme deflections are ineffective when it comes to tracking the setpoint or returning to equilibrium. The case shown in Figures 4.41 through 4.44 is worse still in that the MPC controller, and most likely any other controller, cannot even maintain stability. The stabilator is considered failed at its most downward saturation point, causing the aircraft to pitch over and quickly depart the linear region represented by the state space model.

4.2 Pitch Pointing

Pitch pointing is a maneuver in which the flight path angle and the forward velocity are held constant while the pitch angle tracks a setpoint. For a fighter aircraft, pitch pointing is important in that it provides "point-and-shoot" capability or, in other words, the ability to track a target without altering the flight path of the aircraft. Furthermore, as demonstrated in [4,7], it is a relatively effortless maneuver that does not require a significant amount of control power to accomplish. Similar to the simulations of the vertical translation maneuver, the aircraft is commanded to begin tracking its setpoint at time t = 0 and then commanded to return to its equilibrium condition at t = 5.0. Unless otherwise specified, the tracking weight is $R_y = \text{diag}(10^4,10^4,10)$, and the control weight is $R_u = (10^{-3})S^TS$. The sample time is 0.5 seconds.

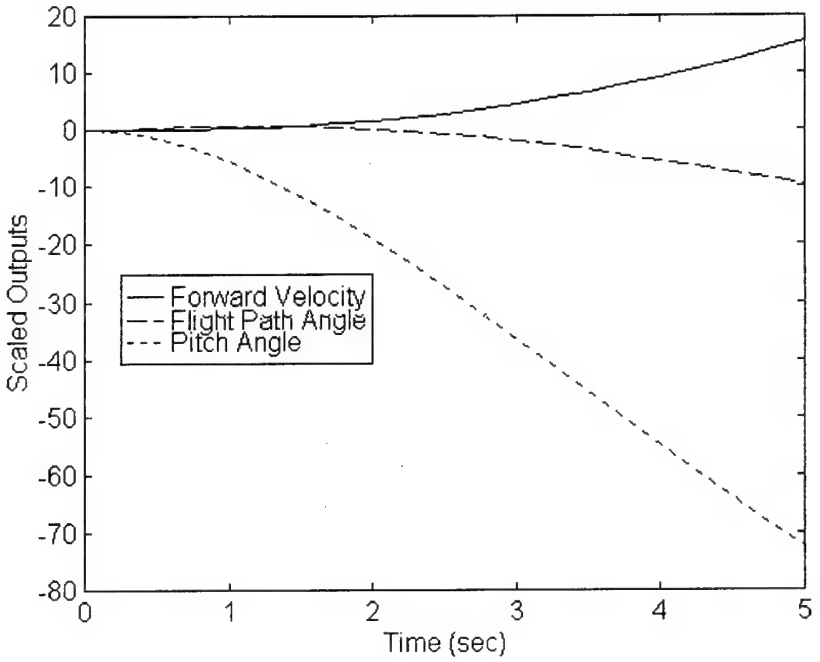


Figure 4.41 VT Output Response (Failure: SS at t=0)

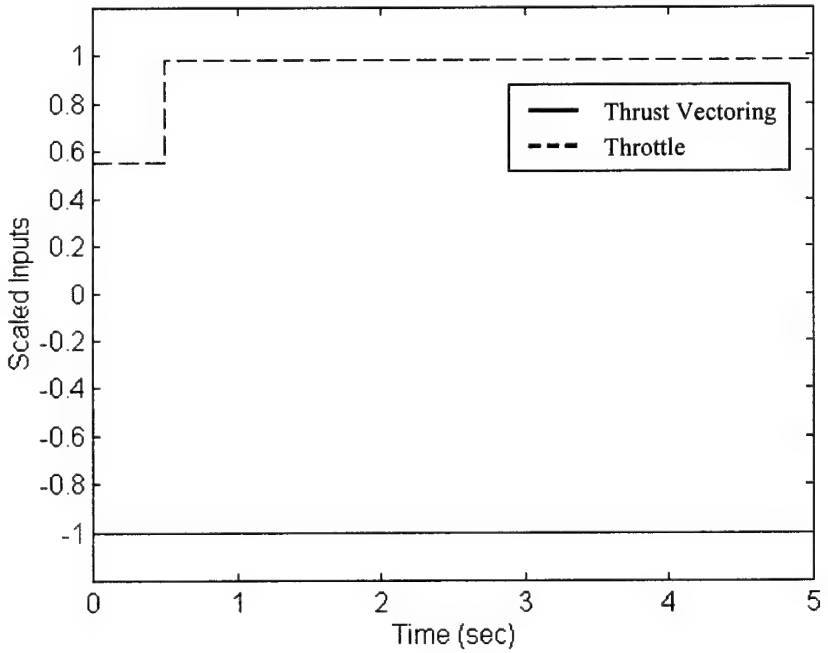


Figure 4.42 VT Control Defl. (Failure: SS at t=0)

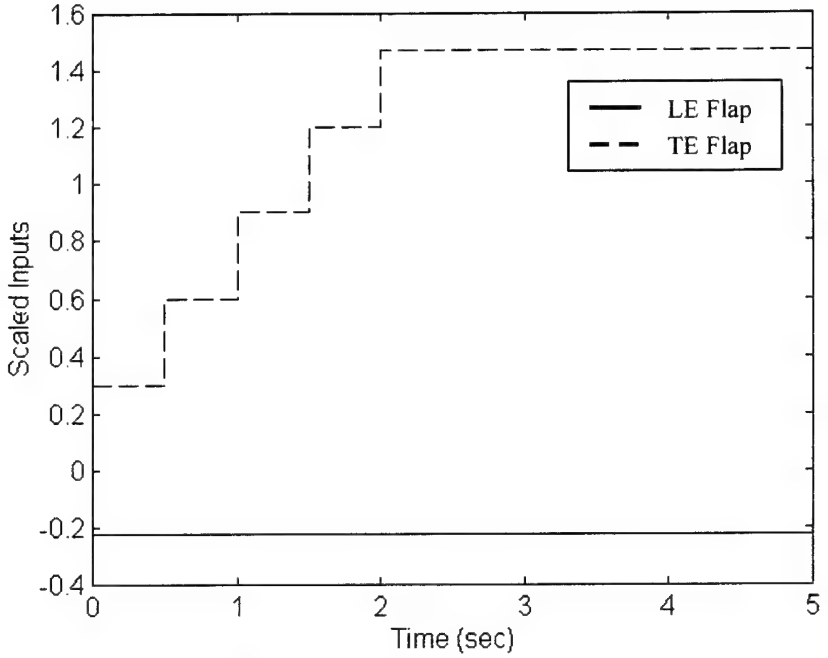


Figure 4.43 VT Control Deflections (Failure: SS at t=0)

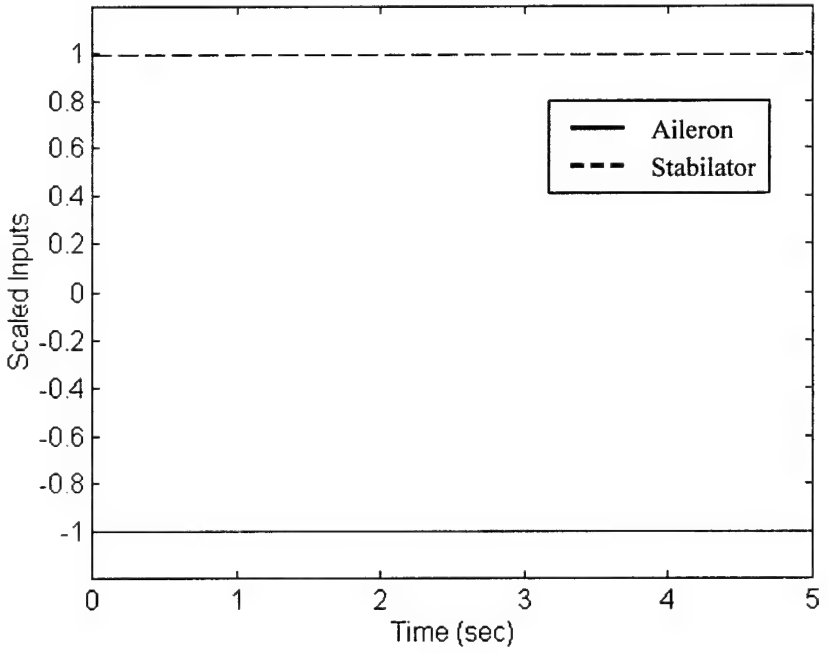


Figure 4.44 VT Control Deflections (Failure: SS at t=0)

The unconstrained pitch pointing maneuver shown in Figures 4.45 through 4.48 demonstrates very clearly that only minimal control usage is necessary to successfully complete this maneuver; the only constraint violation is the rate constraint on the trailing edge flaps. Implementing the constraints, as shown in Figures 4.49 through 4.52, does not significantly affect performance, but does increase overall control usage to achieve the same level of tracking. During the maneuver, the thrust vectoring vanes and the stabilator are primarily responsible for establishing the pitch angle, while the ailerons and trailing edge flaps deflect upward to dissipate the excess lift caused by the increased angle of attack. Similar to the vertical translation maneuver, the thrust vectoring also appears to play a role in minimizing deviations of the forward velocity from its equilibrium value.

Because control deflections are relatively small in this maneuver, additional failure scenarios similar to those shown in the section dealing with vertical translation do not provide additional insight into the problem of dealing with actuator failures. However, the tracking weights chosen for the pitch pointing maneuver do serve to provide a useful example of how weightings chosen for optimal tracking may not be ideally suited to the aircraft following the failure of a control element. For instance, in Figures 4.53 through 4.56 the aircraft suffers a simulated multiple failure of the stabilator at t=0.5 seconds and the ailerons at t=1.0 second. Given the small deflections at which these failures occur, the thrust vectoring vanes and the trailing edge flaps easily compensate for the malfunctions, and the initial setpoint tracking performance is only slightly worse in terms of settling time. Where the difficulty arises is when the aircraft is

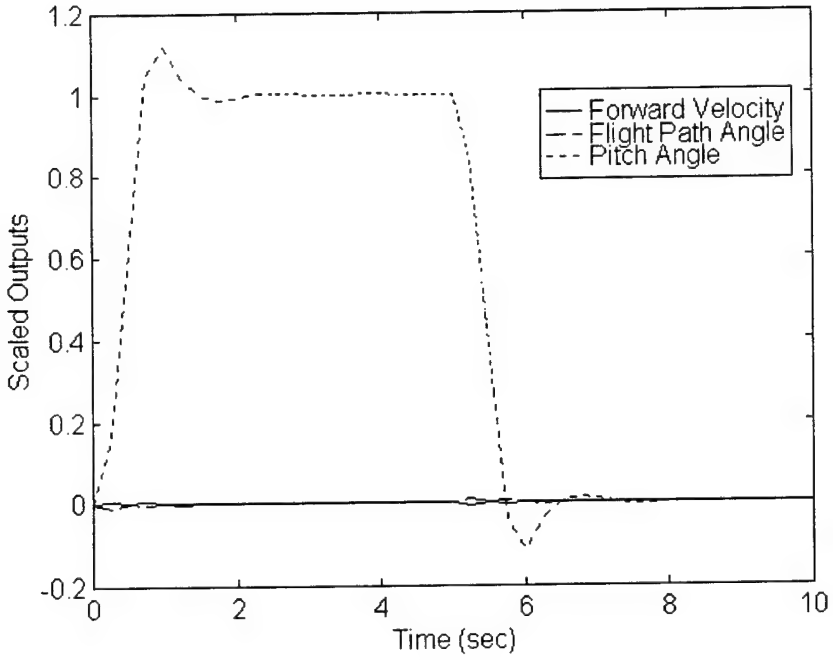


Figure 4.45 PP Output Response (Unconstrained)

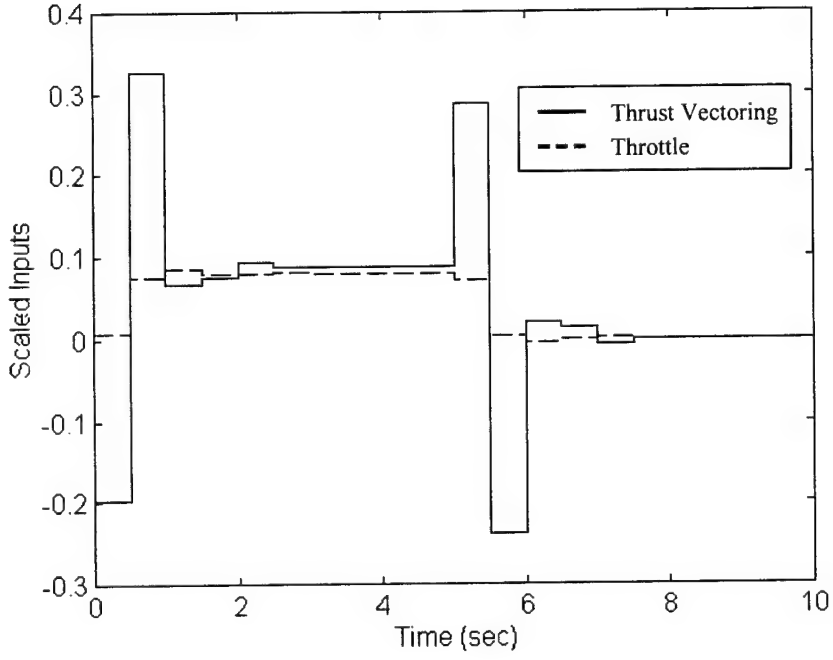


Figure 4.46 PP Control Deflections (Unconstrained)

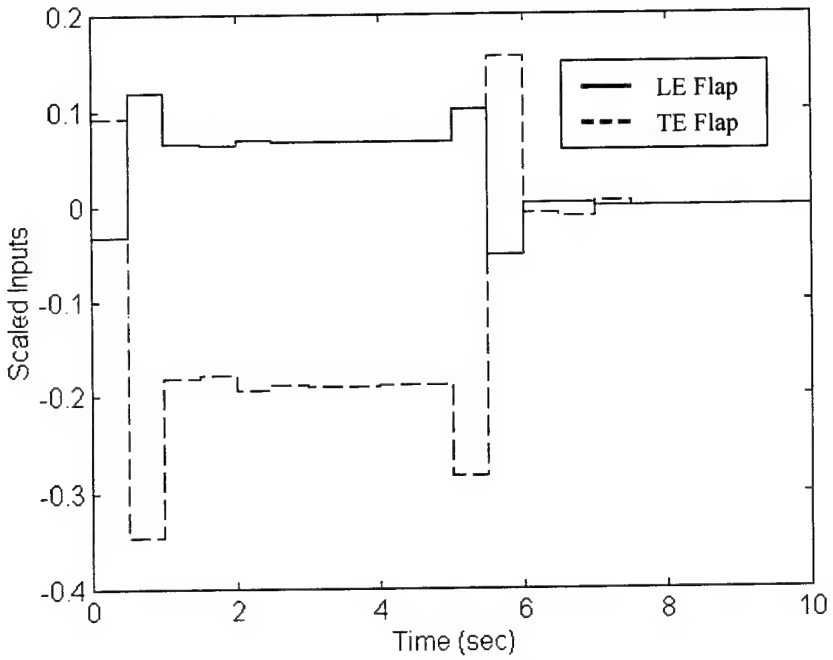


Figure 4.47 PP Control Deflections (Unconstrained)

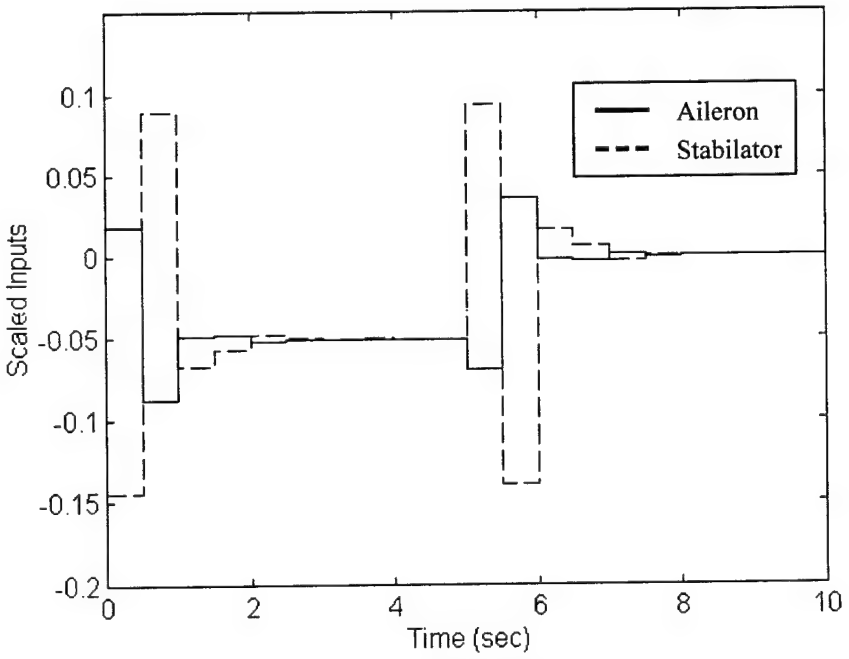


Figure 4.48 PP Control Deflections (Unconstrained)

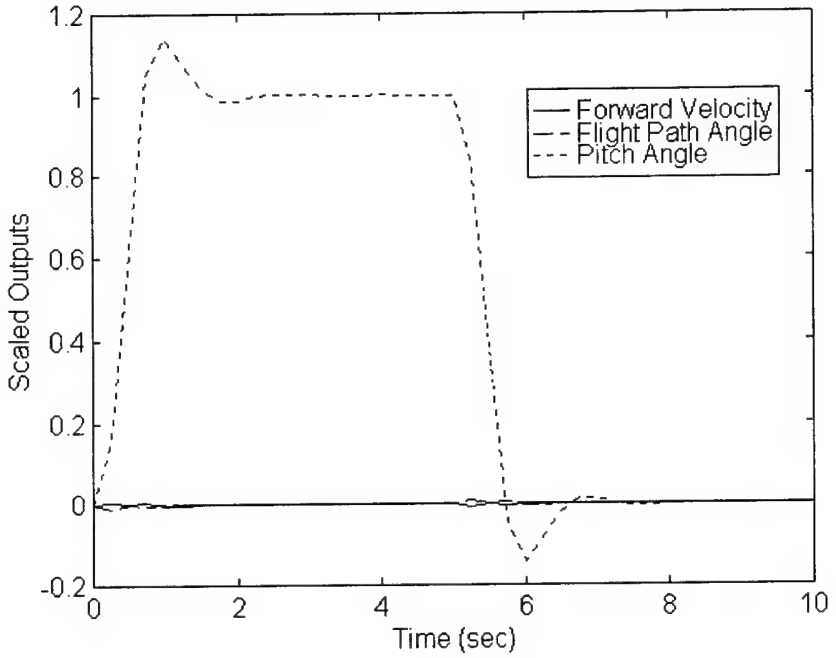


Figure 4.49 PP Output Response (Nominal Constraints)

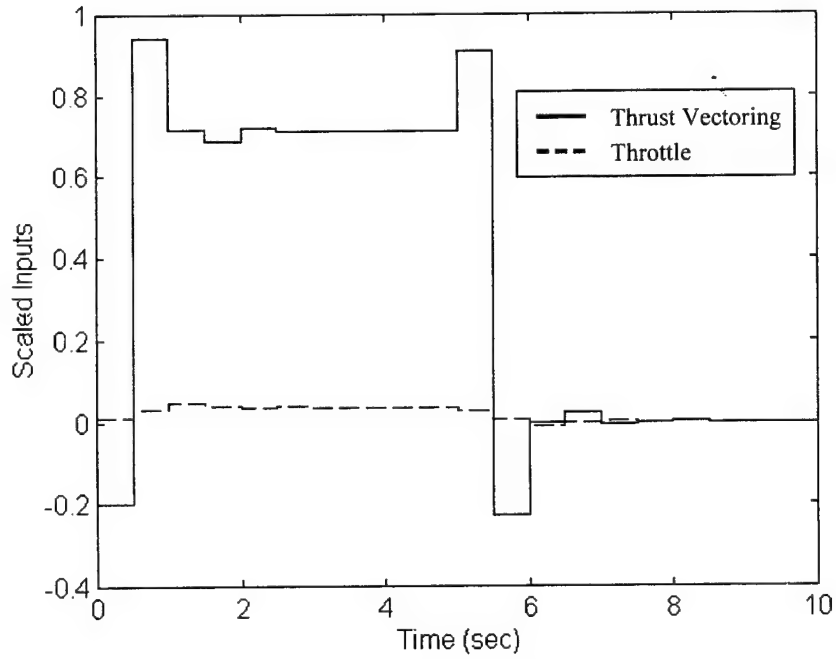


Figure 4.50 PP Output Response (Nominal Constraints)

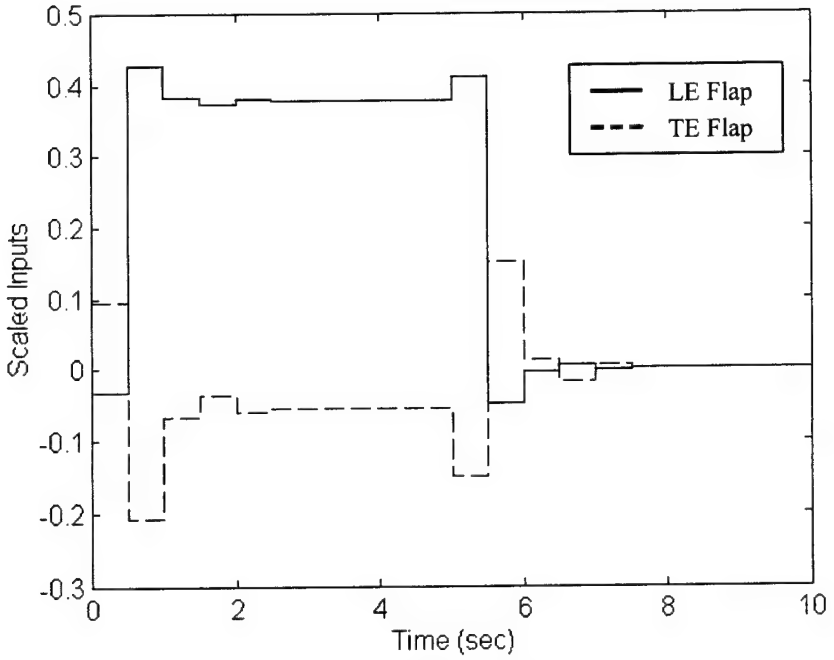


Figure 4.51 PP Control Deflections (Nominal Constraints)

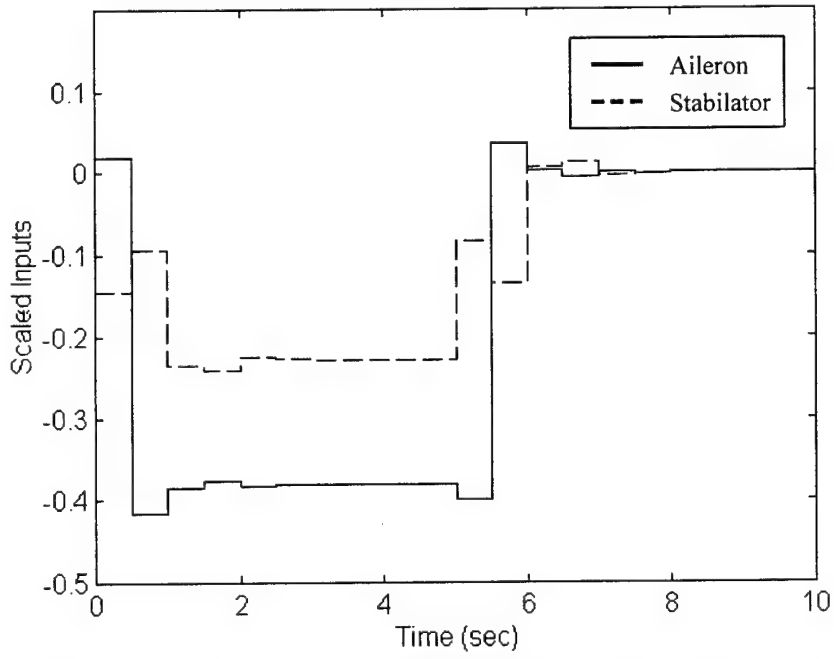


Figure 4.52 PP Control Deflections (Nominal Constraints)

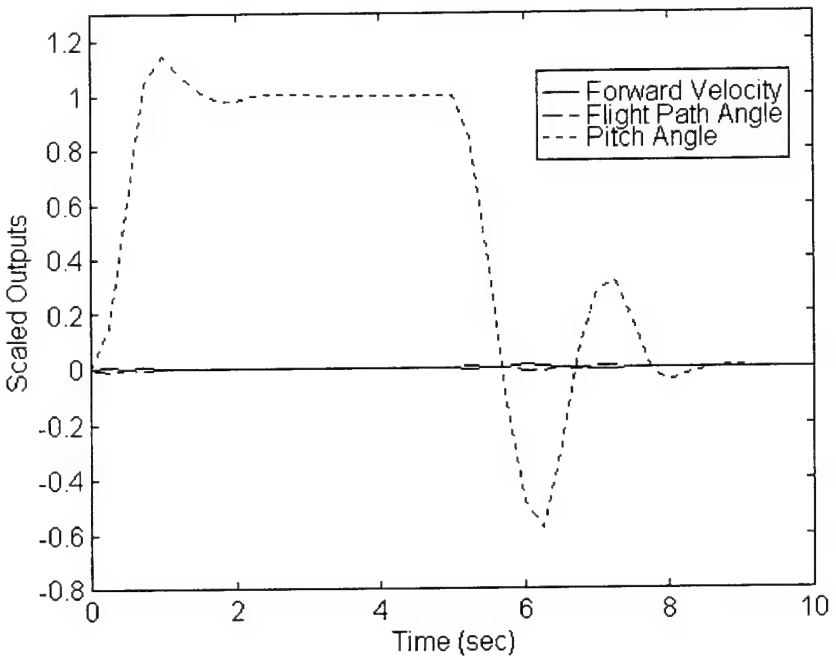


Figure 4.53 PP Output Resp. (Failures: SS at t=0.5; AS at t=1.0)

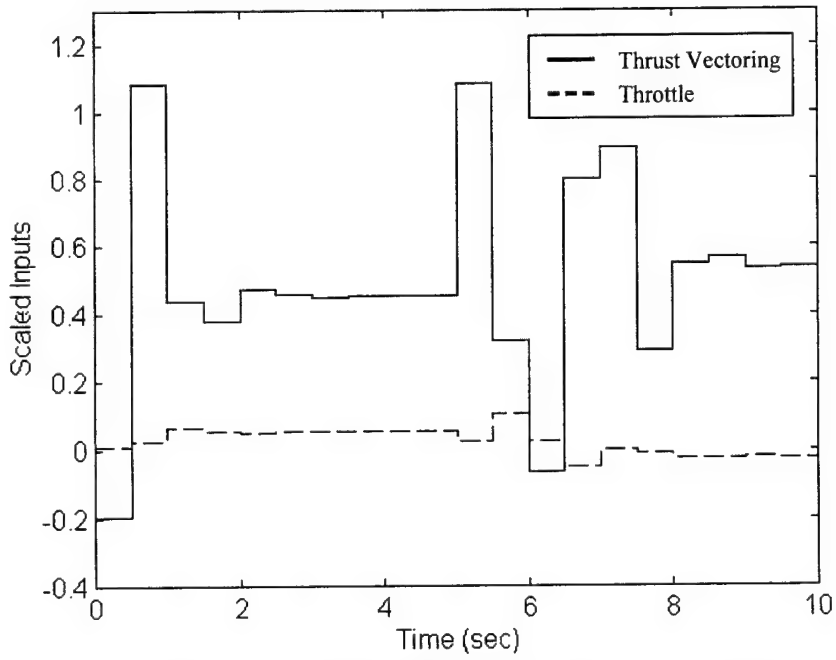


Figure 4.54 PP Control Defl. (Failures: SS at t=0.5; AS at t=1.0)

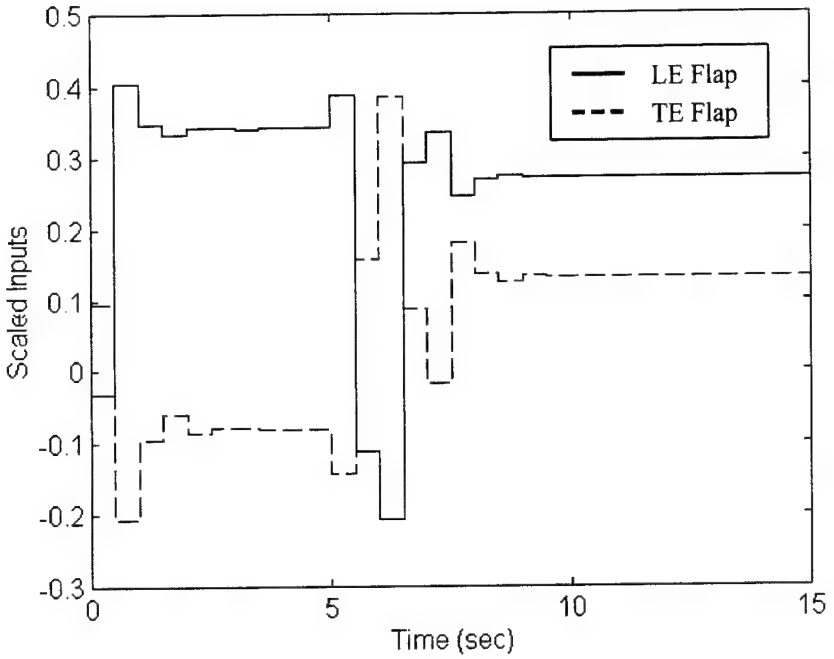


Figure 4.55 PP Control Defl. (Failures: SS at t=0.5; AS at t=1.0)

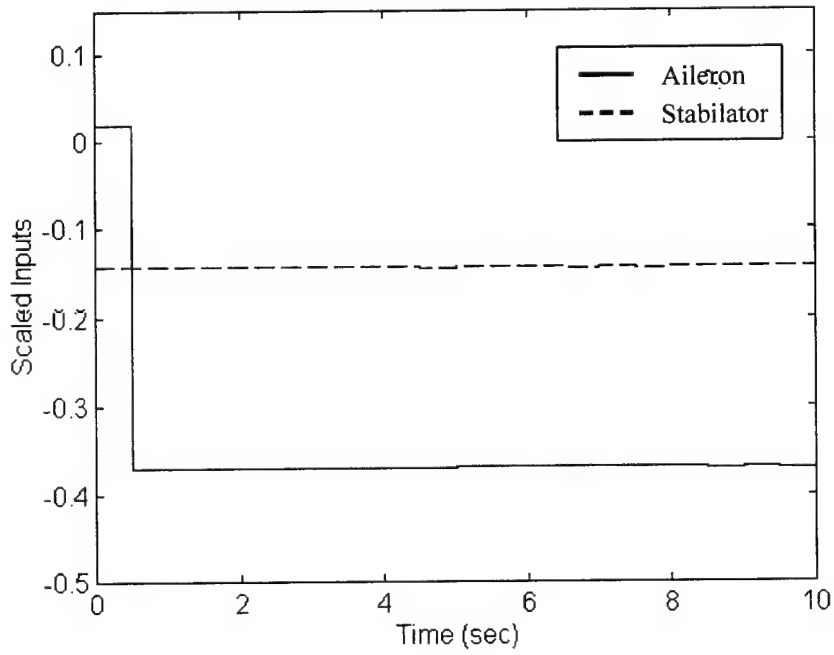


Figure 4.56 PP Control Defl. (Failures: SS at t=0.5; AS at t=1.0)

commanded to return to its original setpoint. The response to this command is oscillatory and undesirable from a handling qualities perspective. The cause of this oscillatory behavior can be linked to the fact that the tracking weights for the forward velocity and flight path angle are three orders of magnitude greater than the tracking weight for the pitch angle. Thus, the pitch angle is given the least priority when it comes to minimizing its perturbations from its designated setpoint. To remedy this situation, we relax the weights on the forward velocity and flight path angle and slightly increase the weight on the pitch angle at t=5.0 seconds: R_y = diag(20,20,4). The effect on the output response is a significant decrease in the magnitude of the oscillations and only a slight increase in the perturbations of the flight path angle and the forward velocity from their trim values. Figures 4.57 through 4.60 contain the control deflections and the output response for this case.

4.3 Direct Lift

The direct lift maneuver is one in which the angle of attack is held constant while the flight path angle and the pitch angle vary. The purpose of the direct lift simulations presented in this thesis is to explore how varying the prediction, control, and optimization horizons affects the performance of the controller. In all cases, the basic stability criterion presented in Section 2.3.1 is used to ensure stability of the resulting closed loop system. The control and tracking weights are selected to be R_y = (25,20,1) and R_u = (1.2e-4) for every simulation. The sample time is t = 0.5 seconds.

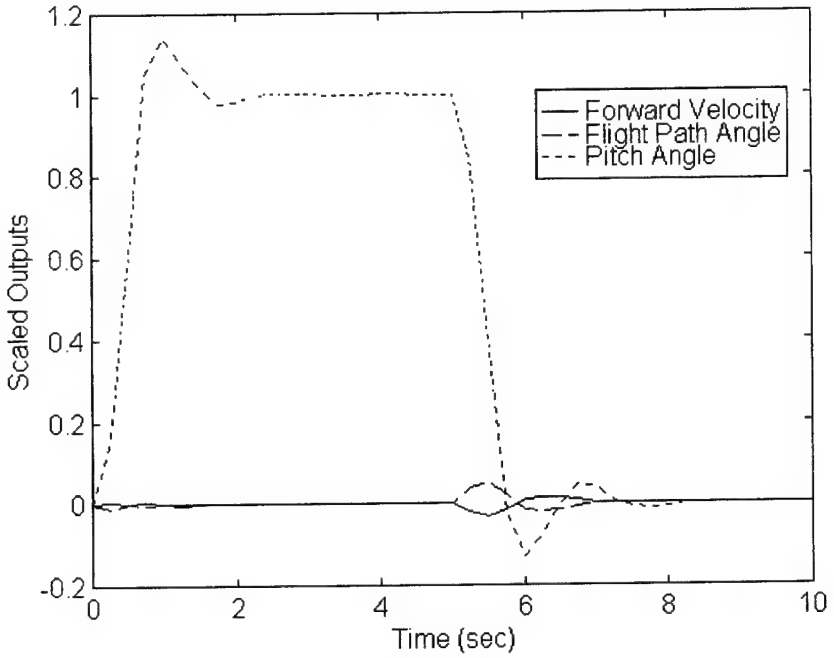


Figure 4.57 PP Output Response (Relaxed Weights)

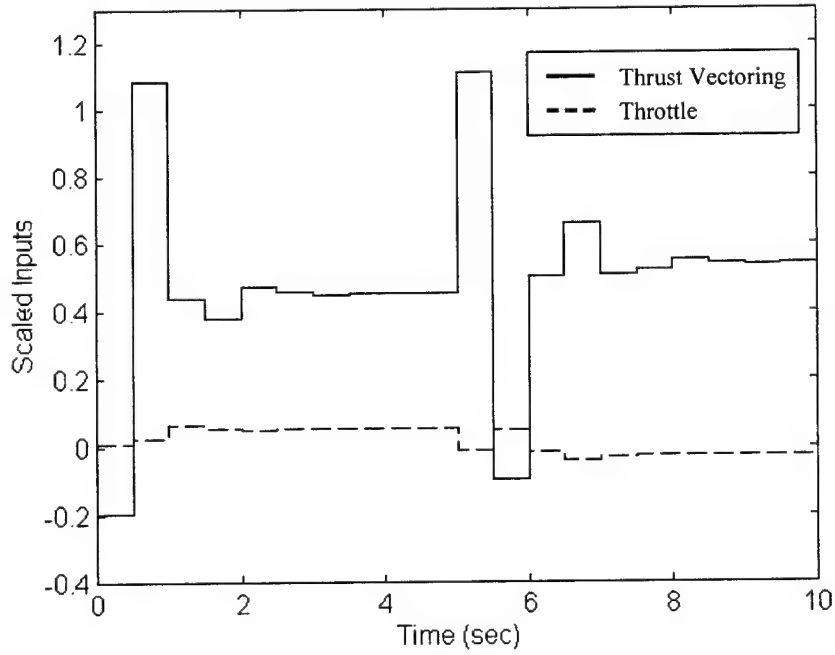


Figure 4.58 PP Control Deflections (Relaxed Weights)

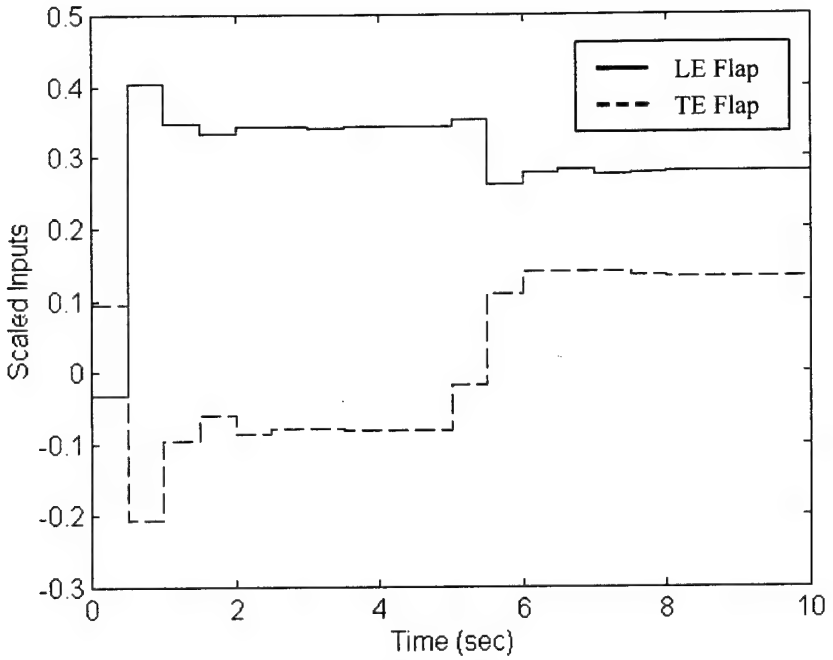


Figure 4.59 PP Control Deflections (Relaxed Weights)

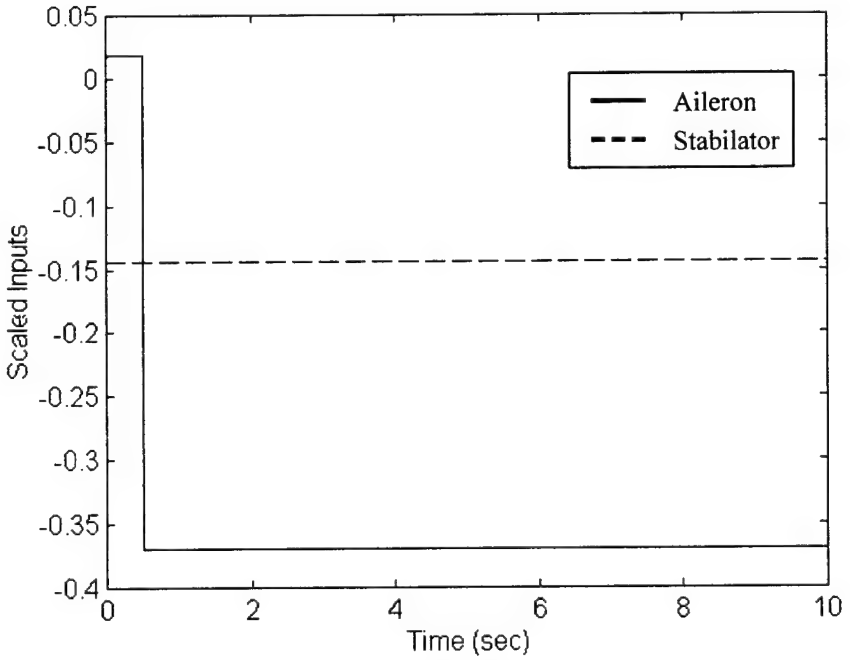


Figure 4.60 PP Control Deflections (Relaxed Weights)

Displayed in Figures 4.61 through 4.64, the first test case employs the standard horizon lengths used throughout this thesis: $r = 5$, $p = q = 20$. Like the vertical translation maneuvers shown in Section 4.1, direct lift requires considerable control power to accomplish. During the initial setpoint tracking, the thrust vectoring vanes reach their upward saturation point, and the ailerons reach their downward saturation point. Additionally, both the trailing edge flaps and the throttle are constrained by their respective rate limits.

Increasing the optimization horizon to $r = 10$ has no discernible effect on system performance. Figures 4.65 through 4.68 indicate that the control deflections and the output response of this case are exactly identical to the nominal case of $r = 5$, $p = q = 20$. Note that ensuring system stability with an optimization horizon increase requires that the prediction and control horizons also be lengthened: $r = 10$, $p = q = 25$. Similarly, Figures 4.69 through 4.72 show that an additional augmentation of the optimization horizon to $r = 15$ is equally ineffective in altering system tracking performance.

The next two scenarios involve fixing the optimization horizon at $r = 5$ and then varying the prediction horizon independently. In Figures 4.73 through 4.76 we see that increasing the prediction horizon to $p = 25$ has no effect on system performance. All control deflections are identical to those of the nominal case, and the output response is equally unaffected. Figures 4.77 through 4.80 show that a further increase of the prediction horizon to $p = 30$ has similarly negligible effects on the system tracking performance.

THIS PAGE LEFT INTENTIONALLY BLANK

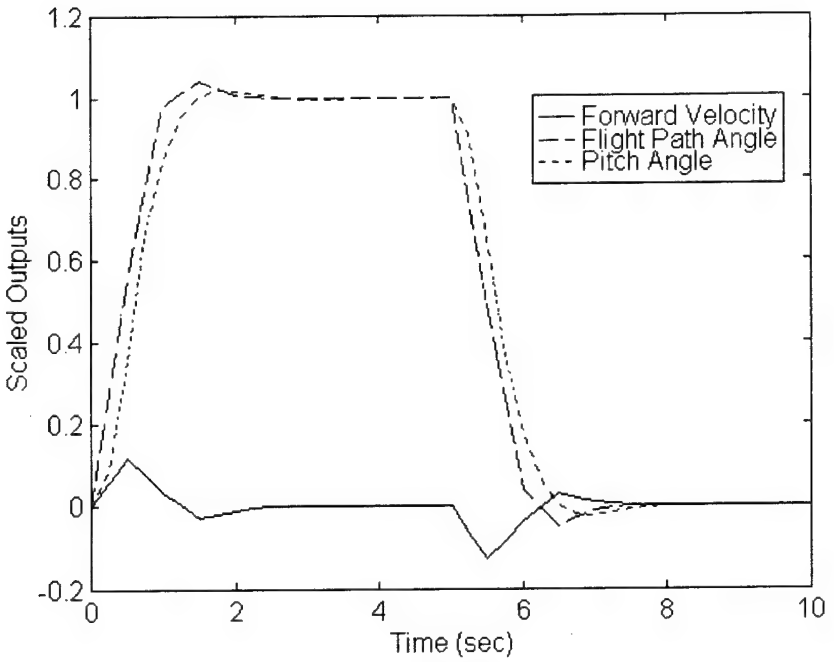


Figure 4.61 DL Output Response (r=5, p = q = 20)

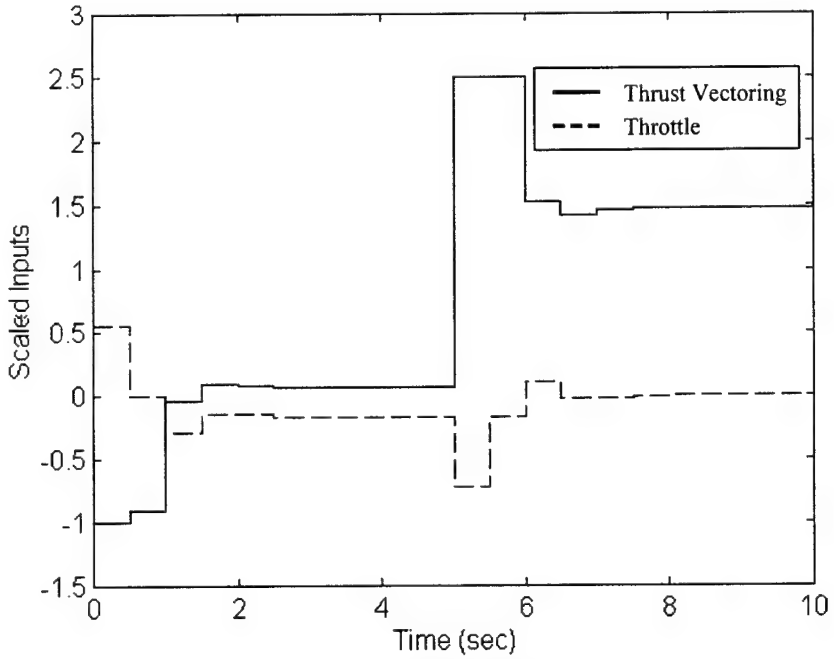


Figure 4.62 DL Control Deflections (r = 5, p = q = 20)

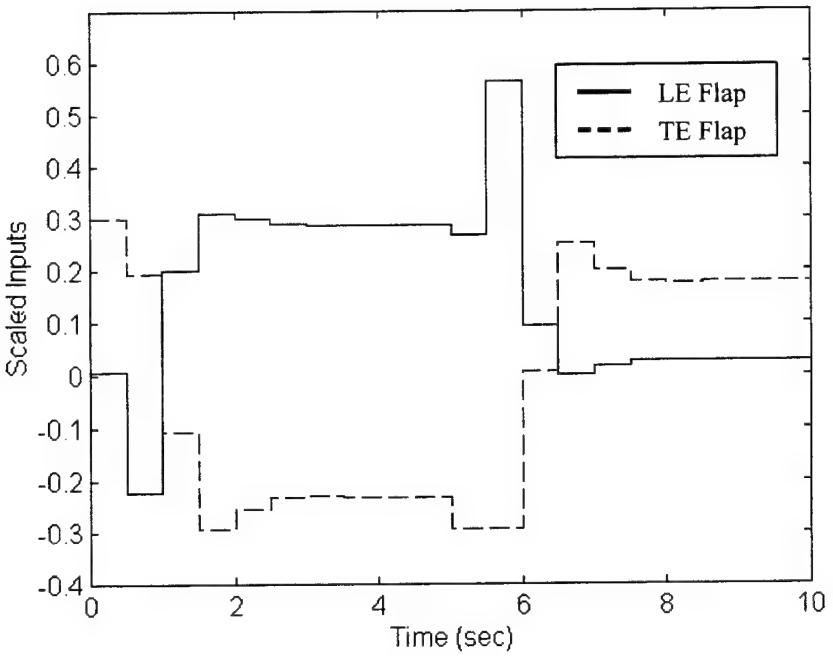


Figure 4.63 DL Control Deflections (r = 5; p = q = 20)

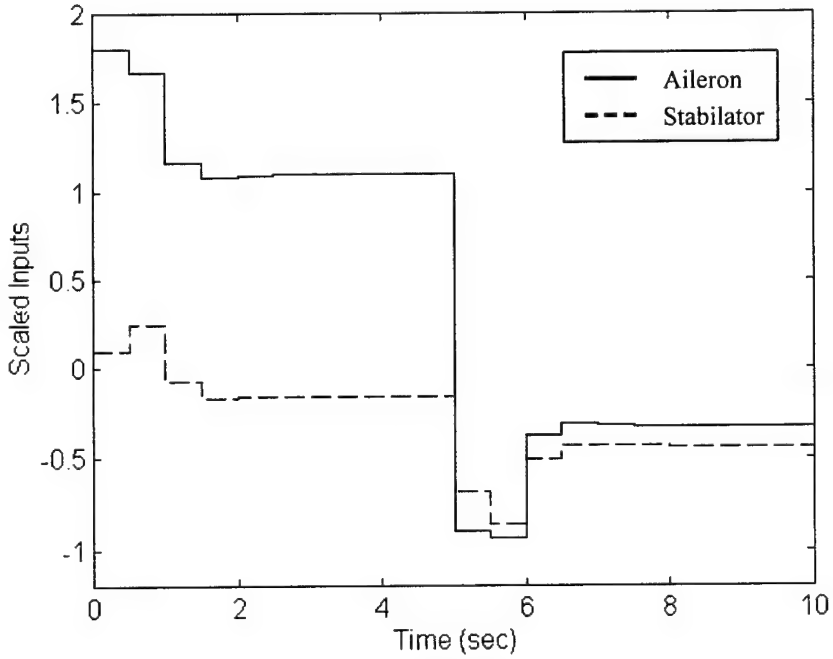


Figure 4.64 DL Control Deflections (r = 5, p = q = 20)

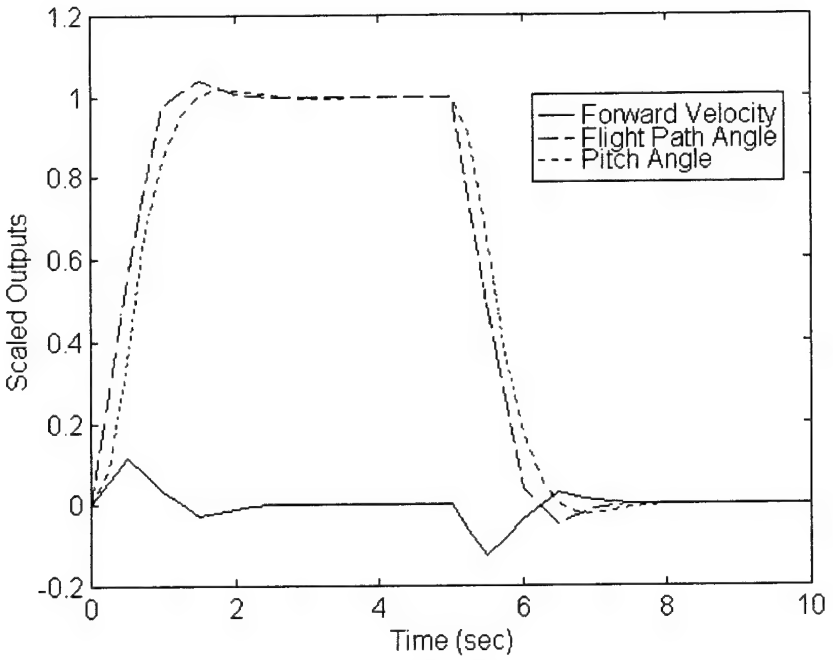


Figure 4.65 DL Output Response (r = 10; p = q = 25)

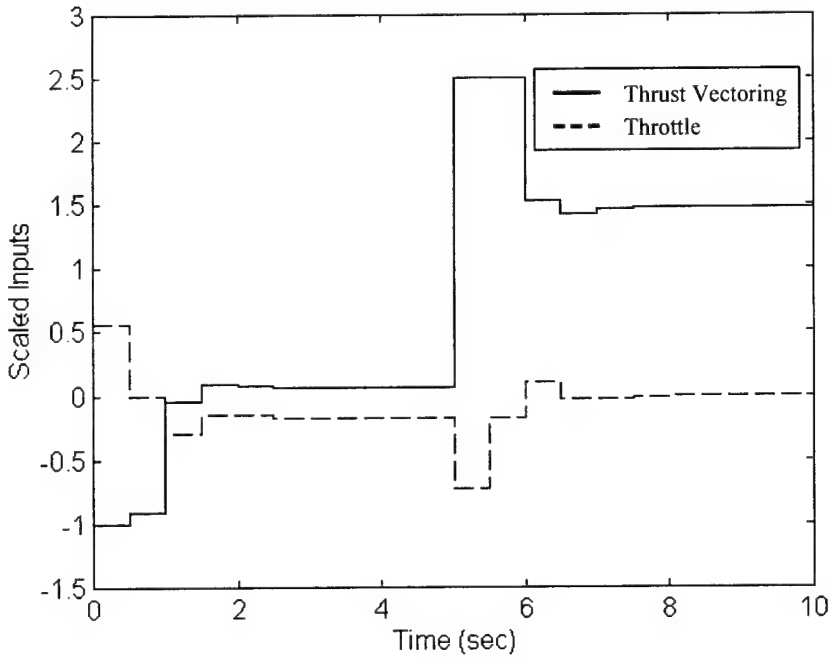


Figure 4.66 DL Control Deflections (r = 10; p = q = 25)

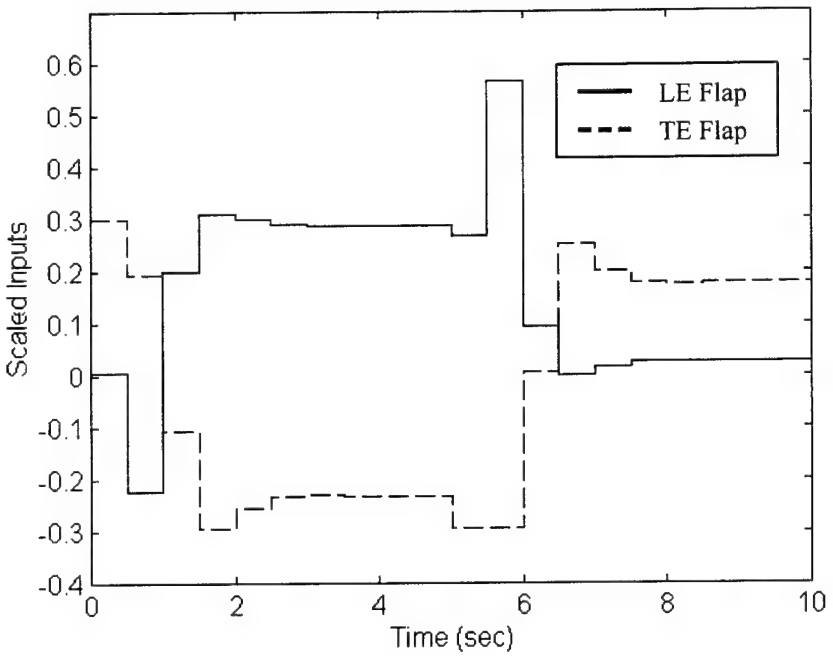


Figure 4.67 DL Control Deflections (r = 10; p = q = 25)

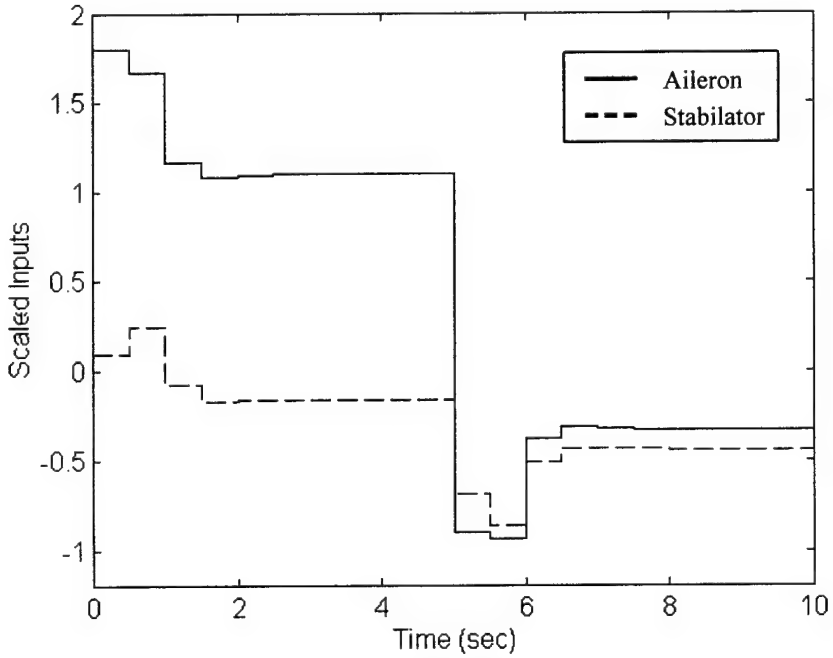


Figure 4.68 DL Control Deflections (r = 10; p = q = 25);

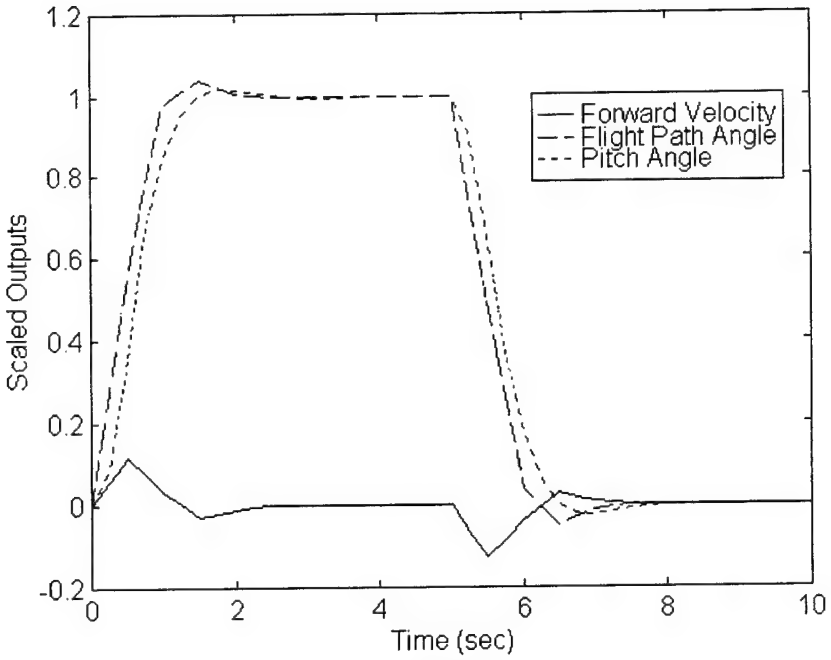


Figure 4.69 DL Output Response (r = 15; p = q = 30)

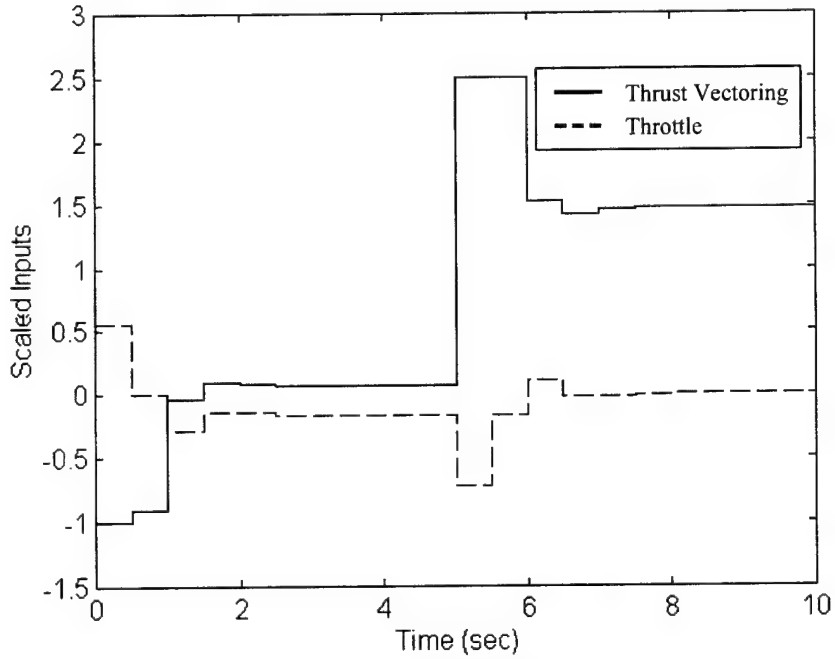


Figure 4.70 DL Control Deflections (r = 15; p = q = 30)

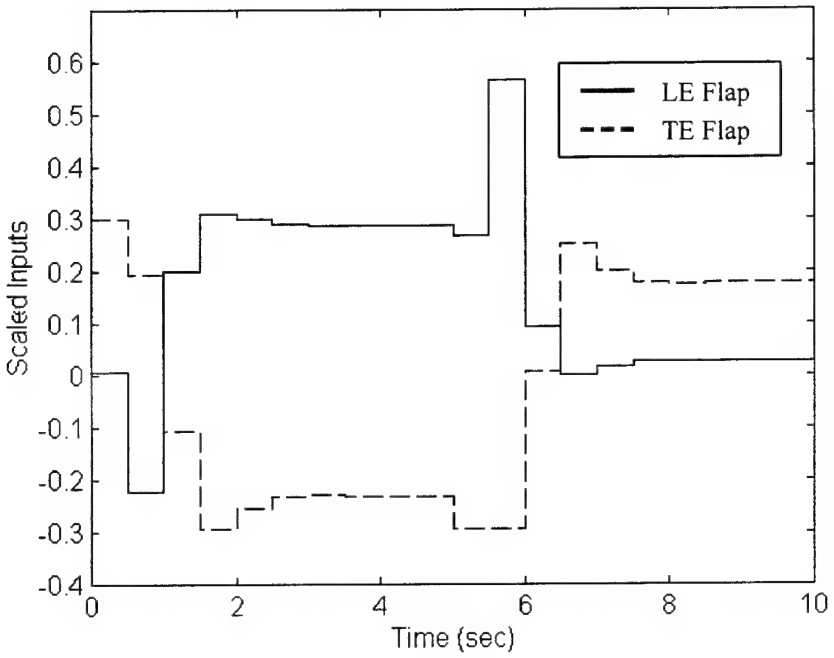


Figure 4.71 DL Control Deflections (r = 15; p = q = 30)

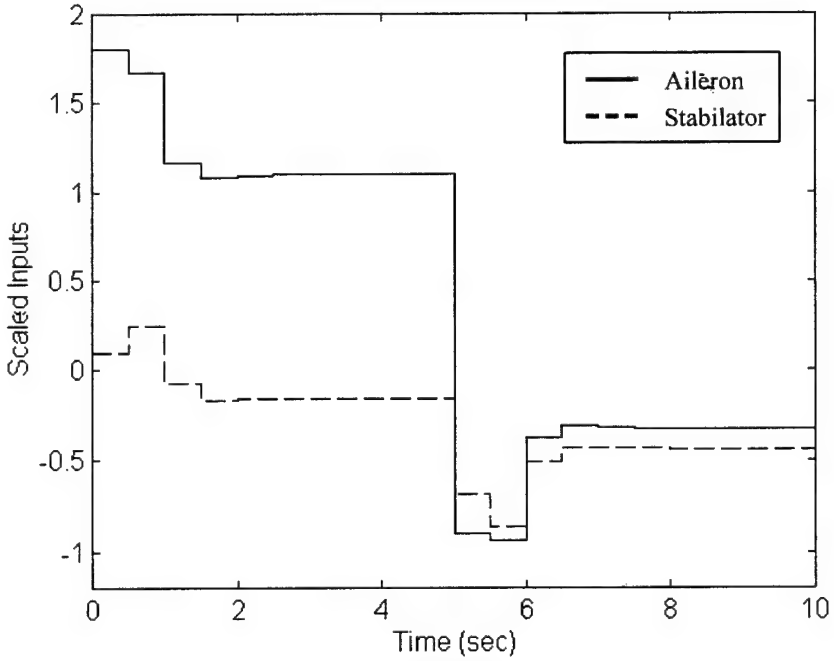


Figure 4.72 DL Control Deflections (r = 15; p = q = 30);

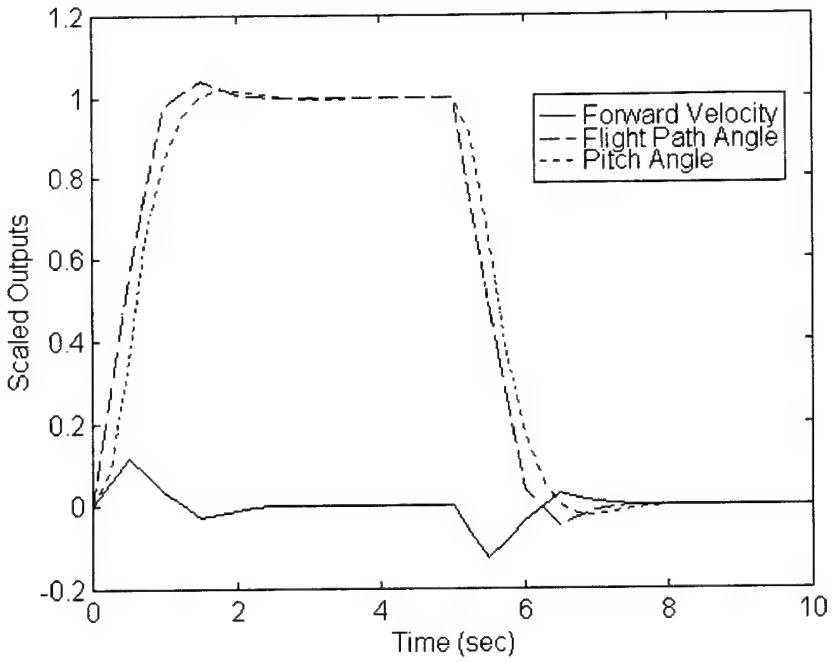


Figure 4.73 DL Output Response (r = 5; p = 25; q = 20)

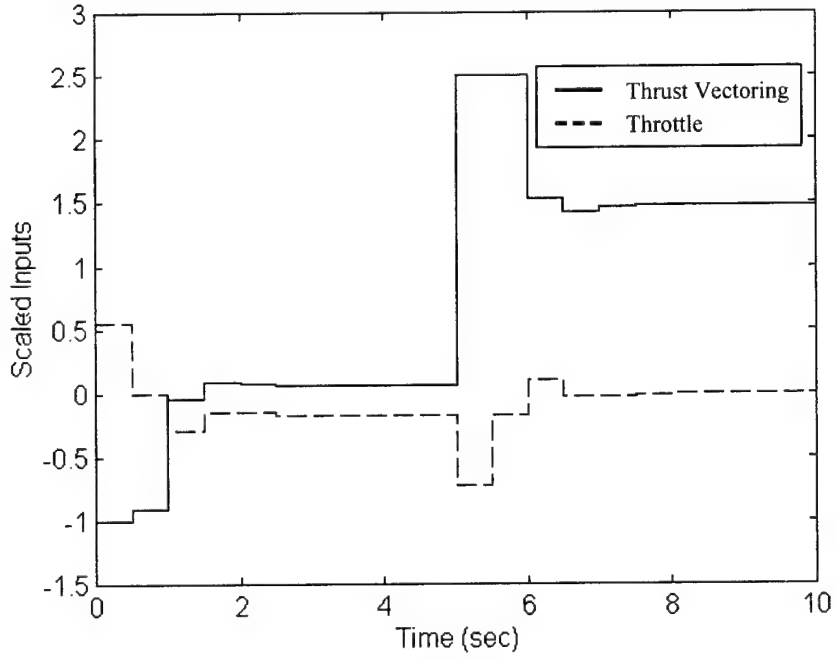


Figure 4.74 DL Control Deflections (r =5; p = 25; q = 20)

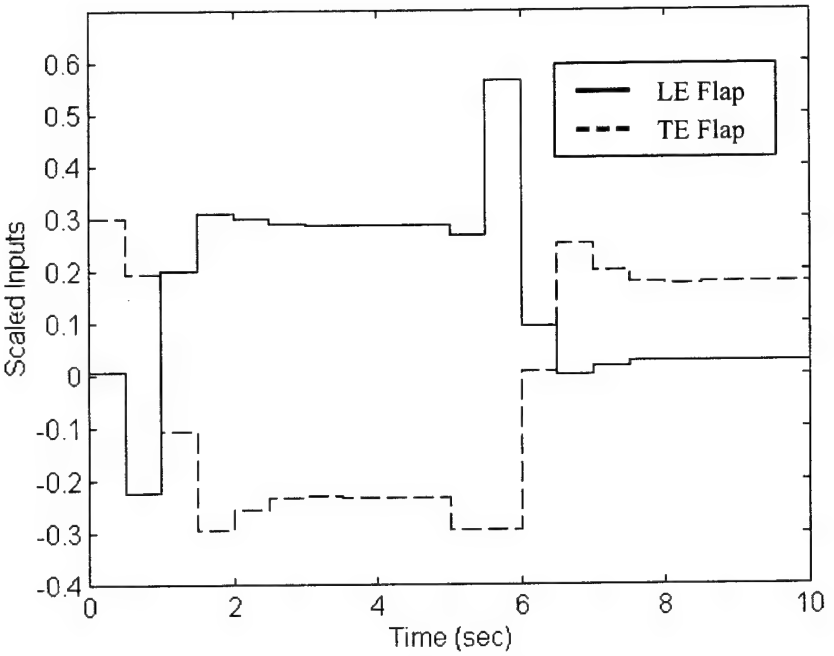


Figure 4.75 DL Control Deflections (r = 5; p = 25; q = 20)

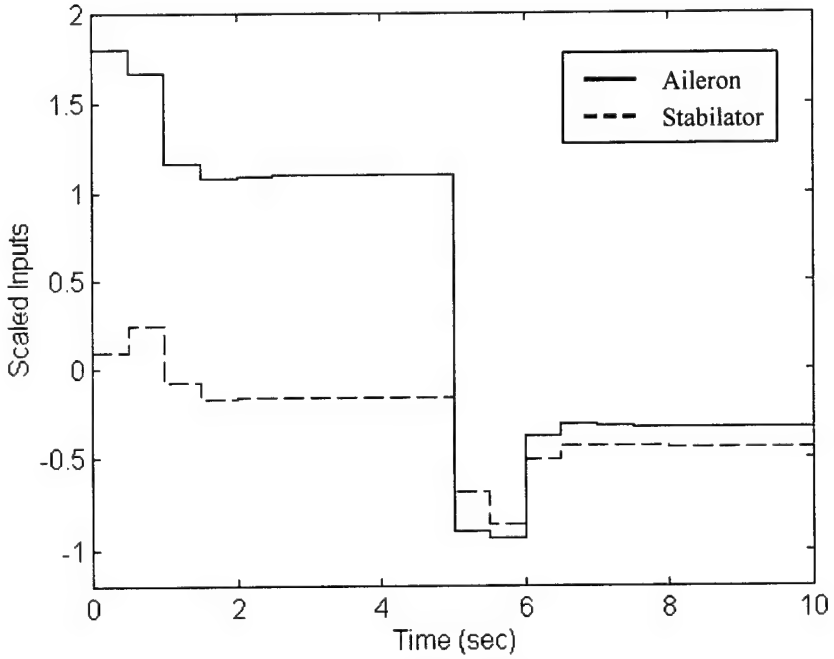


Figure 4.76 DL Control Deflections (r = 5; p = 25; q = 20);

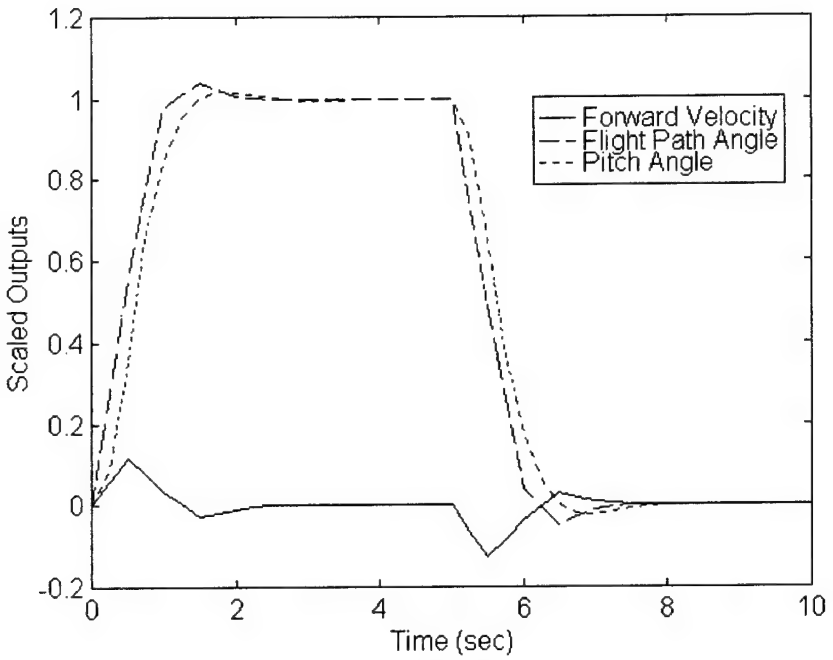


Figure 4.77 DL Output Response (r = 5; p = 30; q = 20)

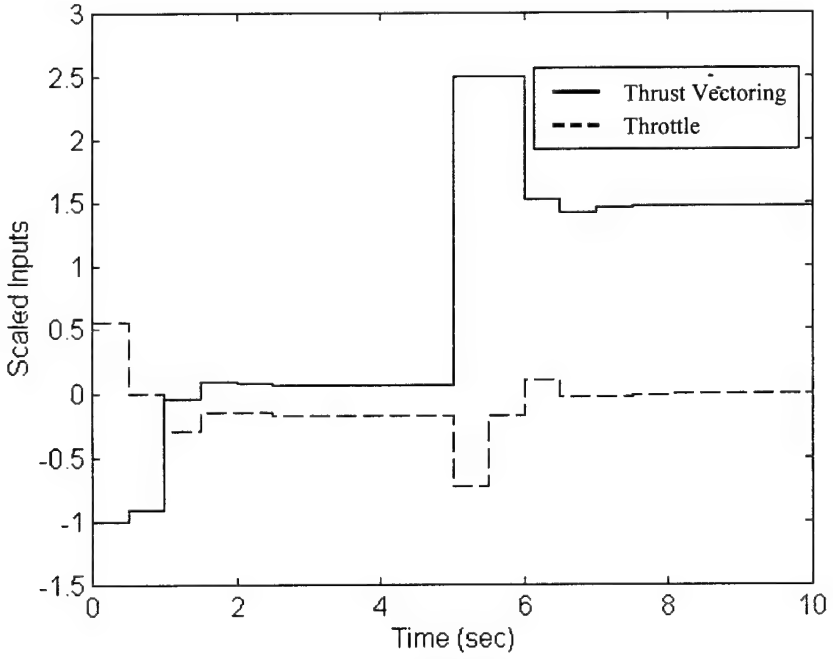


Figure 4.78 DL Control Deflections (r =5; p = 30; q = 20)

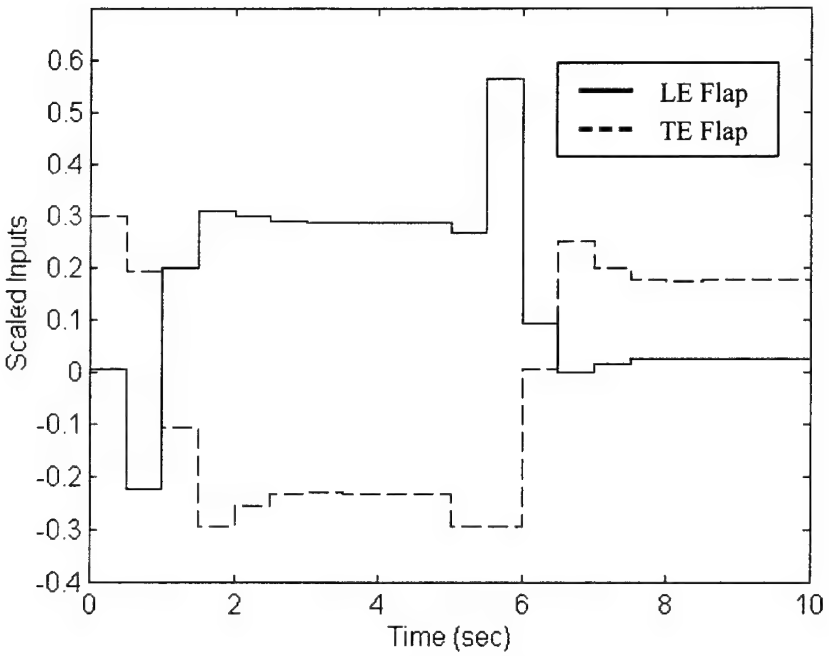


Figure 4.79 DL Control Deflections (r = 5; p = 30; q = 20)

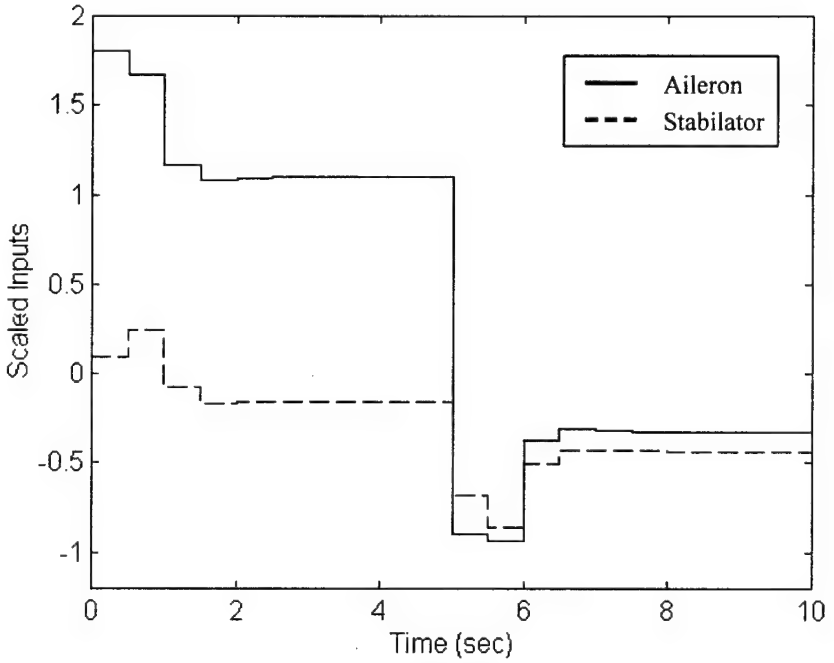


Figure 4.80 DL Control Deflections (r = 5; p = 30; q = 20);

5.0 Conclusions and Directions for Future Research

5.1 Conclusions

As shown in the vertical translation simulations, MPC has the capability to redistribute control power in order to mitigate the effects of single and multiple control surface failures so long as the specified failures are not too extreme. For minor failures, the MPC controller even maintains near-nominal performance. Another interesting result demonstrated in this thesis is how on-line adjustment of the optimization weightings can significantly improve system performance, especially subsequent to a control element failure. Demonstrated in the pitch pointing simulations, the change of weights following a multiple control element failure significantly reduced pitch oscillations when the aircraft was commanded to move from its setpoint to its original equilibrium condition. Thus, the addition of a mechanism for adaptively choosing optimization weightings seems like a logical addition to MPC aircraft controllers. Lastly, the direct lift simulations present a case for choosing the MPC horizons as small as possible because, in this specific instance, increasing them beyond the minimums for system stability only increases calculation time and does not discernibly benefit overall system performance.

5.2 Directions for Future Research

A next logical step in the use of MPC controllers for control of aircraft is to augment the straight four state F-18 linearized model with actuator and sensor dynamics

in the attempt to increase the overall fidelity of the simulation. Furthermore, if possible, outputs fed into the estimator should be those actually available from the F-18 HARV or whatever aircraft is being studied.

In the arena of fault-tolerant control, the pitch pointing simulations indicate that the addition of an adaptive optimization weight selection routine could conceivably improve system performance. Additionally, only actuator failure scenarios and not battle damage scenarios were studied in this thesis because in most cases battle damage would have at least some effect on aircraft dynamics. Thus, the addition of a system identification routine linked to the MPC optimizer would likely enable the MPC controller to update the prediction equations based on the updated plant model and maintain system stability.

Appendix A1: F-18 HARV Unscaled Model

$$\dot{x}_o(t) = A_o x_o(t) + B_o u_o(t)$$
$$y_o(t) = C_o x_o(t)$$

$$A_o = \begin{bmatrix} -7.5000e-2 & -2.4050e+1 & 0 & -3.2160e+1 \\ -8.8250e-4 & -1.9590e-1 & 9.8960e-1 & 0 \\ -1.5000e-4 & -1.4540e-1 & -1.8770e-1 & 0 \\ 0 & 0 & 1.0000e+0 & 0 \end{bmatrix}$$

$$B_o = \begin{bmatrix} -2.2980e-2 & 0 & -7.2865e-2 & 3.9300e-2 & -4.1140e-2 & 1.6000e-1 \\ -1.8000e-4 & -1.3000e-4 & -4.3000e-4 & -4.7000e-5 & -3.0000e-4 & -3.0000e-4 \\ -6.7000e-3 & -7.0000e-4 & -1.2000e-2 & -5.7000e-4 & 7.0000e-4 & 4.7000e-4 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$C_o = \begin{bmatrix} 1.0000e+0 & 0 & 0 & 0 \\ 0 & -1.0000e+0 & 0 & 1.0000e+0 \\ 0 & 0 & 0 & 1.0000e+0 \end{bmatrix}$$

Appendix A2: F-18 HARV Scaled Model

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t)$$

$$A = T^{-1}A_oT \qquad B = T^{-1}B_oS \qquad C = R^{-1}C_oT$$

$$A = \begin{bmatrix} -7.5000e-2 & -4.1975e-1 & 0 & -5.6130e-1 \\ -5.0564e-2 & -1.9590e-1 & 9.8960e-1 & 0 \\ -8.5944e-3 & -1.4540e-1 & -1.8770e-1 & 0 \\ 0 & 0 & 1.0000e+0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} -4.5960e-1 & 0 & -2.4774e+0 & 1.1790e+0 & -2.4684e+0 & 4.3200e+0 \\ -2.0627e-1 & -3.7242e-1 & -8.3767e-1 & -8.0787e-2 & -1.0313e+0 & -4.6410e-1 \\ -7.6777e+0 & -2.0054e+0 & -2.3377e+1 & -9.7976e-1 & 2.4064e+0 & 7.2709e-1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 1.2500e-1 & 0 & 0 & 0 \\ 0 & -1.0000e+0 & 0 & 1.0000e+0 \\ 0 & 0 & 0 & 1.0000e+0 \end{bmatrix}$$

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{57.296} & 0 & 0 \\ 0 & 0 & \frac{1}{57.296} & 0 \\ 0 & 0 & 0 & \frac{1}{57.296} \end{bmatrix} \qquad S = \begin{bmatrix} 20 & 0 & 0 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 & 0 & 0 \\ 0 & 0 & 34 & 0 & 0 & 0 \\ 0 & 0 & 0 & 30 & 0 & 0 \\ 0 & 0 & 0 & 0 & 30 & 0 \\ 0 & 0 & 0 & 0 & 0 & 27 \end{bmatrix}$$

$$R = \begin{bmatrix} 8 & 0 & 0 \\ 0 & \frac{1}{57.296} & 0 \\ 0 & 0 & \frac{1}{57.296} \end{bmatrix}$$

Appendix B1: Example of Controller Setup Script File for Matlab

WRITTEN BY: Derek W. Ebdon

```
% Script file for controller setup.  "s1t1.m"
% VERTICAL TRANSLATION
% CONTROL SURFACES:  No failure
% This problem will use the differencing formulation from equations (2.18) and (2.19)
% to manipulate the plant into a form that accepts control increment inputs.

% The flight condition is as follows:

%    Alt.:  15,000 ft
%    Mach:  0.24
%    Vt  :  238.7 ft/s
%    AOA :  25 deg
%    PA  :  25 deg  (pitch angle)
%    FPA :   0 deg  (flight path angle)

% Define the sample rate of the system

Tsample = 0.5;

% Perform the input-output and state scalings and then discretize the plant.
% The plant states are: Vt, alpha, q, theta
% The plant outputs are:  Vt, gamma, theta

harvop1  % Load the state space longitudinal model and the scaling matrices.

ascl = inv(Ts)*alon*Ts;   % Perform the scalings
bscl = inv(Ts)*blon*Ss;
cscl = inv(Rs)*clon*Ts;
dscl = inv(Rs)*dlon*Ss;

[a,b,c,d] = c2dm(ascl,bscl,cscl,dscl,Tsample,'zoh');  %  Discretize using a zero order hold

% The F-18 HARV has 6 inputs to the system :
%    1) symmetric thrust vectoring vane
%    2) symmetric aileron deflection
%    3) symmetric stabilator deflection
%    4) symmetric leading edge flap
```

```
%    5) symmetric trailing edge flap
%    6) throttle


% Create the state-space matrices for a plant that will accept input
% increments instead of absolute control inputs.

A = [ a zeros(4,3) ; c*a eye(3) ];
B = [ b ; c*b ];
C = [ zeros(3,4) eye(3) ];
D = zeros(3,6);


% Create a continuous time model that accepts control increment inputs for use in
Simulink.

at = [ ascl zeros(4,3) ; cscl*ascl eye(3) ];
bt = [ bscl ; cscl*bscl ];
ct = [ zeros(3,4) eye(3) ];
dt = zeros(3,6);


% Establish the prediction, optimization, and control horizons.

p = 20;          % prediction horizon
q = 20;          % control horizon
r = 5;           % optimization horizon
rho = 20;


% Establish Z matrix weights

Zr = eye(6);
Zl = eye(6);


% Establish R optimization weights

Ry = [ 50 0 0 ; 0 20 0 ; 0 0 1e4 ];
Ru = (1e-6)*Ss'*Ss;
```

```
% Determine the number of control inputs, outputs, and states.

kappa = size(A,1); %estimator states
xi = size(B,2);    %control inputs
eta = size(C,1);   %system outputs


% Establish the setpoint if not performed in Simulink.

% sinf = [0 1 0]';


% Establish the constraint matrices: Ll, Mm, Nn.  These matrices implement the
% constraints at time k only.

Ll = [ -eye(xi) zeros(xi,xi*(q-1)) ; eye(xi) zeros(xi,xi*(q-1)) ];
Mm = [ -eye(xi) zeros(xi,xi*(q-1)) ; eye(xi) zeros(xi,xi*(q-1)) ];
Nn = [];


% Establish physical constraint limits: l, m, n
% The rate constraints assume a 0.5 second sample time.

l =    [ 4.0 2.0 1.176 0.500 0.300 0.556  4.0 2.0 1.176  0.500 0.300 0.556 ]';
m =    [ 1.0 1.0 1.035 0.222 0.293 1.722  2.5 1.8 0.994  2.178 1.473 0.981 ]';
n =    [];

l2 =   [ 4.0 2.0 1.176 0.500 0.300 0.556  4.0 2.0 1.176  0.500 0.300 0.556 ]';
m2 =   [ 1.0 1.0 1.035 0.222 0.293 1.722  2.5 1.8 0.994  2.178 1.473 0.981 ]';
n2 =   [];

l3 =   [ 4.0 2.0 1.176 0.500 0.300 0.556  4.0 2.0 1.176  0.500 0.300 0.556 ]';
m3 =   [ 1.0 1.0 1.035 0.222 0.293 1.722  2.5 1.8 0.994  2.178 1.473 0.981 ]';
n3 =   [];


% Calculate the gains, Fr and L, that place the poles of the plant and the estimator at the
% origin.

[Fr,L]=gains(A,B,C,D);
```

```
% Calculate the state space matrices for the stabilizing inner loop controller.

[Ax,Bx,Cx,Dx,Ay,By,Cy,Dy,F0,F1,G,H]=controller(A,B,C,Fr,L,Zl,Zr);


% Formulate the prediction matrices.

[cF,cG,cGinf,cH,cFdel,cGdel,cGdelinf,cHdel,cFu,cGu,cGuinf,cHu]=
        pmats(kappa,xi,eta,p,q,r,rho,F0,F1,G,H,Fr,Zr);


% Calculate the matrices necessary to perform the quadratic programming optimization.

[S,cB,cD,cE]=
        qpconstants(C,Ry,Ru,cF,cG,cGinf,cH,cFdel,cGdel,cHdel,cGdelinf,cFu,cGu,cHu,
        cGuinf,Ll,Mm,Nn,p,q,r,rho);


% Calculate the vector of constraints across the MPC horizons.

[ll,mm,nn,ll2,mm2,nn2,ll3,mm3,nn3]=constraint(l,m,n,l2,m2,n2,l3,m3,n3,p,q);


% Calculate setpoint trajectory and final value of "quasi-reference" signal, v.

%[vinf] = vinfy(A,B,C,Fr,L,Zr,sinf,r,rho);
%[s] = setpoint(sinf,p);


save A A
save B B
save C C
save Fr Fr
save L L
save Zr Zr
save rho rho
```

Appendix B2: Kucera Algorithm for Pole Placement

WRITTEN BY: Sharon A. Heise

```
function [Fr,G]=gains(A,B,C,D);

% to make A+BFr and A+LC stable
% Q=C'*Ry*C;
% [Fr,ricsol,evals1]=dlqr(A,B,Q,Ru);
% Fr=-Fr;
% [L]=dlqe(A,eye(size(A)),C,eye(size(A)),eye(size(C,1)));
% L=-L;
% to place poles of A+BFr and A+LC at zero

%
% just to remember the original a,b,c,d
%
ra=A;
rb=B;
rc=C;
rd=D;

kappa=size(A,1);
xi=size(B,2);
emp=eye(xi);

debug=0;

X=[];
Y=[];
Yn=[];
XY=[];

if (rank(ctrb(A,B))~=kappa)
  error('[A,B] is not controllable');
end

%
% since a*a*a is more accurate than a^3
% use Anb=A^i * B
Anb=B;
```

```
for i=1:kappa;

    %
    % kucera's C_i.
    %

    %disp(sprintf('Iteration start: i=%d',i));

    XYA=[XY Anb];


        %
        % The trouble with having a full rank test like this
        % is that it is possible to add _too_many_ columns
        % to XY using D_1, so that there is no room to make
        % D_2 etc non-zero. So lets just skip it and add only
        % one column maximum.
        %


                Yn=[];
                testXY=XY;
                foundone=0;

                for j=1:size(Anb,2)
                        newcol=Anb(:,j);
                        if ( rank([testXY newcol]) == size([testXY newcol],2) )
                                testXY=[testXY newcol];
                                foundone=1;
                                Yn=[Yn  emp(:,j)] ;
                                %
                                % Note that, in order to not make
                                % D_1 (say) very wide, and then
                                % have to make D_2,3,4 etc all 0,
                                % we insert a break here so that
                                % we stop searching after finding one rank
                                % addition
                                %
                                disp(sprintf('breaking at i=%d j=%d',i,j));
                                break;
                        end
                end
```

```
                if (foundone==0)
                        XY
                        Anb
                        error(sprintf('Failed to find a D_%d',i));
                end


    XY=[XY Anb*Yn]

    place(i)=size(Y,2)+1;
    Y=[Y Yn];

    X=[X Anb*Yn];
    place1(i)=size(X,2);

    %
    % update Anb
    %

    Anb=A *  Anb;

end;

place(i+1)=size(Y,2)+1;
place1(i+1)=size(X,2);



if (debug~=0)

                disp(sprintf('place='));
                place

                disp(sprintf('place1='));
                place1

                disp(sprintf('Rank of [X1Y1 X2Y2 ...] = %d',rank(X)));

                %
                % D_i is given by Y(:,place(i):place(i+1)-1)
                %
```

```
% C_i is given by A^(i-1)*B
%
% [C_1D_1 C_2D_2 ... C_iD_i] is given by
%
%      X(:,1:place1(i))
%


%
% some test code.
%
%

disp('-----------------------------------------------');
for i=1:kappa

%
% kucera's C1
%
  Anb=A^(i-1)*B;
  disp(sprintf('Kuceras C_%d=',i));
        Anb

%
% kucera's D1
%
  myd=Y(:,place(i):place(i+1)-1);
  disp(sprintf('Kuceras D_%d=',i));
        myd

%
% [C1D1 C2D2 ... CiDi]
%
  disp(sprintf('[C1D1 C2D2 ... CiDi]='));
  myx=X(:,1:place1(i))
  disp(sprintf('with rank=%d',rank(myx)));

disp('-----------------------------------------------');

end

end
```

```
L=zeros(xi,kappa);


for i=1:kappa;

  %
  % first find myx=[C1D1 C2D2 ... CiDi]
  %
  temp1=X(:,1:place1(i));

  %
  %  all we have to do to get temp2 is copy the matrix Y up to D_i
  %  and zero out its first few columns.
  %

  temp2=Y(:,1:place(i+1)-1);
  temp2(: , 1:place(i)-1) = zeros(size(temp2( : , 1:place(i)-1) ));

  %
  %   Lbar=temp2*pinv(temp1)
  %
  %  we are really just solving linear equations here ; far better
  %  to use / than pseudo inverse. also faster. see "help slash"
  %

  Lbar = temp2/temp1;

  %
  % slow but accurate way to compute (a-b*l) ^ k

  pm=A-B*L;
  pmr=pm;

  if i>1
    for j=1:i-1
      pmr=pmr * pm;
    end
  end

  L = L + Lbar * pmr;
end;
Fr=-L;
```

```
%
% test
%

eig(A-B * L)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% And now we try for the observer...
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



A=A';
B=C';

kappa=size(A,1);
xi=size(B,2);
emp=eye(xi);

debug=0;

X=[];
Y=[];
Yn=[];
XY=[];

if (rank(ctrb(A,B))~=kappa)
   error('[A,C] is not observable');
end

place=[];
place1=[];
```

```
%
% since a*a*a is more accurate than a^3
% use Anb=A^i * B

Anb=B;

for i=1:kappa;

    %
    % kucera's C_i.
    %

    %disp(sprintf('Iteration start: i=%d',i));

    XYA=[XY Anb];


            Yn=[];
            testXY=XY;
            foundone=0;

            for j=1:size(Anb,2)
                    newcol=Anb(:,j);
                    if ( rank([testXY newcol]) == size([testXY newcol],2) )
                            testXY=[testXY newcol];
                            foundone=1;
                            Yn=[Yn emp(:,j)];
                            break;
                    end
            end

            if (foundone==0)
                    error(sprintf('Failed to find a D_%d',i));
            end


    XY=[XY Anb*Yn];

    place(i)=size(Y,2)+1;
    Y=[Y Yn];
    X=[X Anb*Yn];
    place1(i)=size(X,2);
```

```
    %
    % update Anb
    %

    Anb=A *  Anb;

end;

place(i+1)=size(Y,2)+1;
place1(i+1)=size(X,2);


if (debug~=0)

                disp(sprintf('place='));
                place


                disp(sprintf('place1='));
                place1


                disp(sprintf('Rank of [X1Y1 X2Y2 ...] = %d',rank(X)));

                %
                % D_i is given by Y(:,place(i):place(i+1)-1)
                %
                % C_i is given by A^(i-1)*B
                %
                % [C_1D_1 C_2D_2 ... C_iD_i] is given by
                %
                %       X(:,1:place1(i))
                %


                %
                % some test code.
                %
                %

                disp('-------------------------------------------');
```

```
            for i=1:kappa

            %
            % kucera's C1
            %
              Anb=A^(i-1)*B;
              disp(sprintf('Kuceras C_%d=',i));
                    Anb

            %
            % kucera's D1
            %
              myd=Y(:,place(i):place(i+1)-1);
              disp(sprintf('Kuceras D_%d=',i));
                    myd

            %
            % [C1D1 C2D2 ... CiDi]
            %
              disp(sprintf('[C1D1 C2D2 ... CiDi]='));
              myx=X(:,1:place1(i))
              disp(sprintf('with rank=%d',rank(myx)));

            disp('-----------------------------------------------');

            end

end

L=zeros(xi,kappa);

for i=1:kappa;

  %
  % first find myx=[C1D1 C2D2 ... CiDi]
  %
  temp1=X(:,1:place1(i));

  %
  % all we have to do to get temp2 is copy the matrix Y up to D_i
  % and zero out its first few columns.
  %
```

```
  temp2=Y(:,1:place(i+1)-1);
  temp2(: , 1:place(i)-1) = zeros(size(temp2( : , 1:place(i)-1) ));



  %
  %   Lbar=temp2*pinv(temp1)
  %
  %  we are really just solving linear equations here ; far better
  %  to use / than pseudo inverse. also faster. see "help slash"
  %

  Lbar = temp2/temp1;


  %
  % slow but accurate way to compute (a-b*l) ^ k

  pm=A-B*L;
  pmr=pm;

  if i>1
    for j=1:i-1
      pmr=pmr * pm;
    end
  end

  L = L + Lbar * pmr;

end;


G=-L';

%
% test
%

eig(ra + G * rc)
```

Appendix B3: Inner Feedback Loop Controller

WRITTEN BY: Sharon A. Heise

```
function [Ax,Bx,Cx,Dx,Ay,By,Cy,Dy,F0,F1,G,H]=controller(A,B,C,Fr,L,Zl,Zr);

% Controller

Ax=A+L*C;
Bx=L;
Cx=inv(Zl)*Fr;
Dx=zeros(size(Cx,1),size(Bx,2));

Ay=A+B*Fr+L*C;
By=B*Zl;
Cy=Fr;
Dy=Zl;

% State Predictor

% Ap=(A+B*Fr+L*C);
% Bp=[B*Zr;-L];
% Cp=eye(size(Ap,1));
% Dp=zeros(size(Cp,1),size(Bp,2));

F0=(A+B*Fr+L*C);
F1=(A+B*Fr);

%for i=1:p-1;
%  F=[F;A+B*Fr];
%end;

G=B*Zr;
H=-L;
```

Appendix B4: Function to Construct Prediction Matrices

WRITTEN BY: Sharon A. Heise
MODIFIED BY: Derek W. Ebdon

```
function [cF,cG,cGinf,cH,cFdel,cGdel,cGdelinf,cHdel,cFu,cGu,cGuinf,cHu]=
        pmats3(kappa,xi,eta,p,q,r,rho,F0,F1,G,H,Fr,Zr);

% Note this is for constant F
% State prediction equations

%************State Equation Prediction Matrices**********************

cF=[];          F̃
for i=1:p;
 cF=[cF;F1^(i-1)*F0];
end;

cG=zeros(p*kappa,r*xi);      G̃
for i=1:p;
 for j=1:r;
  if i >= j;
   cG((i-1)*kappa+1:i*kappa,(j-1)*xi+1:j*xi) = F1^(i-j)*G;
  end;
 end;
end;

cGinf=zeros(p*kappa,(rho-r)*xi);      G̃^∞
for i=1:p;
 for j=1:rho-r;
  if i-r >= j;
   cGinf((i-1)*kappa+1:i*kappa,(j-1)*xi+1:j*xi)= F1^(i-r-j)*G;
  end;
 end;
end;


cH=[];          H̃
for i=1:p;
 cH=[cH;F1^(i-1)*H];
end;
```

```
%************Control Increment Prediction Matrices**********************

cFdel=[Fr];      F̃_Δ
for i=1:q-1;
 cFdel=[cFdel;Fr*F1^(i-1)*F0];
end;

cGdel=zeros(q*xi,r*xi);        G̃_Δ
for i=1:q;
 for j=1:r;
  if i==j;
   cGdel((i-1)*xi+1:i*xi,(j-1)*xi+1:j*xi)=Zr;
  elseif i > j;
   cGdel((i-1)*xi+1:i*xi,(j-1)*xi+1:j*xi)=Fr*F1^(i-j-1)*G;
  end;
 end;
end;

cGdelinf=zeros(q*xi,(rho-r)*xi);       G̃_Δ^∞
for i=1:q;
 for j=1:rho-r;
  if i-r==j;
   cGdelinf((i-1)*xi+1:i*xi,(j-1)*xi+1:j*xi)=Zr;
  elseif i-r>j;
   cGdelinf((i-1)*xi+1:i*xi,(j-1)*xi+1:j*xi)=Fr*F1^(i-j-r-1)*G;
  end;
 end;
end;

cHdel=[];       H̃_Δ
for i=1:q-1;
 cHdel=[cHdel;Fr*F1^(i-1)*H];
end;
cHdel=[zeros(xi,eta);cHdel];
```

```
%*******************Input Prediction Matrices**********************

cFu=[];          F̃_u
for i=1:q;
 temp=zeros(size(Fr*F0));
 for j=1:i;
  temp=temp+cFdel((j-1)*xi+1:j*xi,:);
 end;
 cFu=[cFu;temp];
end;


cGu=zeros(q*xi,r*xi);          G̃_u
for i=1:q;
 for j=1:r;
  if i==j;
   cGu((i-1)*xi+1:i*xi,(j-1)*xi+1:j*xi)=Zr;
  elseif i>j;
   temp=zeros(size(F0));
   for k=j:i-1;
    temp=temp+F1^(k-j);
   end;
   cGu((i-1)*xi+1:i*xi,(j-1)*xi+1:j*xi)=Fr*temp*G+Zr;
  end;
 end;
end;


cGuinf=zeros(q*xi,(rho-r)*xi);          G̃_u^∞
for i=1:q;
 for j=1:rho-r;
  if i-r==j;
   cGuinf((i-1)*xi+1:i*xi,(j-1)*xi+1:j*xi)=Zr;
  elseif i-r>j;
   temp=zeros(size(F1));
   for k=j:i-r-1;
    temp=temp+F1^(k-j);
   end;
   cGuinf((i-1)*xi+1:i*xi,(j-1)*xi+1:j*xi)=Fr*temp*G+Zr;
  end;
 end;
end;
```

```
cHu=[];          H̃_u
for i=1:q-1;
  temp=zeros(size(F1));
  for j=0:i-1;
    temp=temp+F1^(j);
  end;
  cHu=[cHu;Fr*temp*H];
end;
cHu=[zeros(xi,eta);cHu];
```

Appendix B5: Function to Calculate Quadratic Programming Matrices

WRITTEN BY: Sharon A. Heise
MODIFIED BY: Derek W. Ebdon

```
function [S,cB,cD,cE]=qpconstants(C,Ry,Ru,cF,cG,cGinf,cH,cFdel,cGdel,
        cHdel,cGdelinf,cFu,cGu,cHu,cGuinf,Ll,Mm,Nn,p,q,r,rho);


% Determine the number of inputs, the number of outputs, the number of estimator states,
% and the number of constraints acting along the prediction and control horizons.

eta=size(C,1);
kappa=size(C,2);
xi=size(Ru,1);
lambda=size(Ll,1);
mu=size(Mm,1);
nu=size(Nn,1);

% Formulate the weighting matrices and the matrix with p output matrices along the
% diagonal.

for i=1:p;
 cC((i-1)*eta+1:i*eta,(i-1)*kappa+1:i*kappa)=C;     C̃
 cQ((i-1)*eta+1:i*eta,(i-1)*eta+1:i*eta)=Ry;        R̃_y
end;

for i=1:q;
 cR((i-1)*xi+1:i*xi,(i-1)*xi+1:i*xi)=Ru;            R̃_u
end;

for i=1:q;
 Iq=[Iq;eye(xi)];        Ĩ
end;


% Construct the matrices needed for the quadratic optimization
% Optimization matrices:

S=cG'*cC'*cQ*cC*cG+cGdel'*cR*cGdel;    S
```

```
cB = 2*([cC*cF cC*cH cC*cGinf -eye(p*eta)]'*cQ*cC*cG +
        [cFdel cHdel cGdelinf zeros(q*xi,p*eta)]'*cR*cGdel);        T

% Constraint matrices for the quadratic optimization.  Note constraints only applied to
% inputs:

cD = [ Ll*cGdel ; Mm*cGu ];          D̃

cE=[-Ll*cFdel -Ll*cHdel -Ll*cGdelinf zeros(lambda,xi) eye(lambda) zeros(lambda,mu) ;
        -Mm*cFu -Mm*cHu -Mm*cGuinf -Mm*Iq zeros(mu,lambda) eye(mu) ];      Ẽ

xietkarp=[xi eta kappa r p];


% Save matrices for use with Simulink.

save  S  S
save cB cB
save cD cD
save cE cE
save xietkarp xietkarp
```

Appendix B6: Function to Assign Constraints

WRITTEN BY: Derek W. Ebdon

```
% This function is for establishing constraints when three cases of constraints are
% specified.  Failures may be simulated by assigning a different set of constraints over
% each time period.

function [ll,mm,nn,ll2,mm2,nn2,ll3,mm3,nn3]=constraint(l,m,n,l2,m2,n2,l3,m3,n3,p,q);

% Constraints will only be applied at time k for this thesis, so all that is required is to
% assign one set of constraints for each failure scenario.  If constraints are to be applied
% across the entire control and prediction horizons, we must use loops to assign the
% selected constraint condition to every time step along the respective horizons.
% Remember that the minumum limits are all in the first half of the constraint vector
% (ll,mm,nn) and that all maximum limits are in the second half because of the method in
% which we formulated the constrained quadratic optimization.

ll = l;
mm = m;
nn = n;

ll2 = l2;
mm2 = m2;
nn2 = n2;

ll3 = l3;
mm3 = m3;
nn3 = n3;

% Save constraint vectors for use with Simulink

save ll ll
save mm mm
save nn nn
save ll2 ll2
save mm2 mm2
save nn2 nn2
save ll3 ll3
save mm3 mm3
save nn3 nn3
```

Appendix B7:
Function to Calculate Far Future Value of Quasi-Reference Signal
Function to Establish Setpoint Trajectory

WRITTEN BY: Sharon A. Heise

Quasi-Reference Signal Calculation

```
function [vinf]=vinfy(A,B,C,Fr,L,Zr,sinf,r,rho);

kappa=size(A,1);

X = [A+B*Fr];

temp = zeros(kappa,kappa);
for i=1:2*kappa;
  temp = temp+X^(i-1);
end;

vinf = (C*temp*B*Zr)\sinf;

for i=1:rho-r;
  vinfer = [vinfer;vinf];
end;

vinf = vinfer;
save vinf vinf
```

Setpoint Trajectory

```
function [s]=setpoint(sinf,p);

s=[];

for i=1:p;
  s=[s;sinf];
end;

save s s
save sinf sinf
```

Appendix B8: Simulink Optimization Function

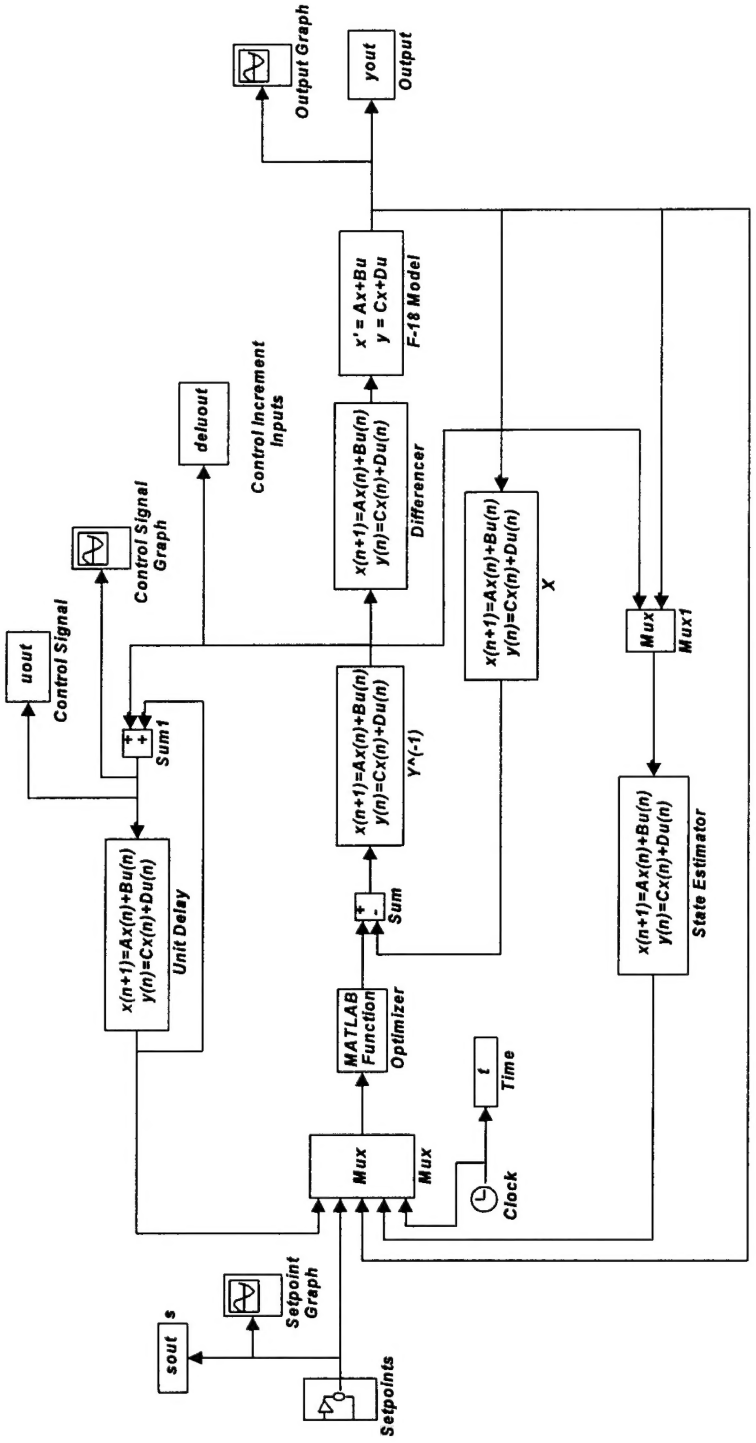
WRITTEN BY: Derek W. Ebdon

```
% This m-file performs a constrained quadratic optimization to find the
% quasi-reference signal v(k) that is input to the plant.

function [v] = optimize(c);

% Load the optimization matrices.  S2 and cB2 are used if the weighting matrices are
% modified to improve performance following an actuator failure.

load  S
load cB
load cD
load cE
%load  S2
%load cB2

% Load the constraint vectors.

load xietkarp
load ll
load mm
load nn
load ll2
load mm2
load nn2
load ll3
load mm3
load nn3

% Load matrices required to determine vinf (far future value of v(k)).

load A
load B
load C
load Fr
load L
load Zr
load rho
```

```
% Determine the number of system inputs, system outputs, estimator states, and the
% lengths of the prediction and optimization horizons.

xi = xietkarp(1);
eta = xietkarp(2);
kappa = xietkarp(3);
r = xietkarp(4);
p = xietkarp(5);

% Accept input from Simulink in order to formulate optimization problem.

u = c(1:xi);
sinf = c(xi+1:xi+eta);
y = c(xi+eta+1:xi+eta+eta);
xhat = c(xi+eta+eta+1:xi+eta+eta+kappa);
t = c(xi+eta+eta+kappa+1);

% Calculate far future value of v(k) and the setpoint trajectory.

[vinf] = vinfy(A,B,C,Fr,L,Zr,sinf,r,rho);
[s] = setpoint(sinf,p);

% Linear term in the quadratic programming problem.  Used just to clean up presentation.
% T1 and T2 represent different weights: Ry and Ru (T1); Ry2 and Ru2(T2).

T1 = [ xhat' y' vinf' s' ]*cB;
T2 = [ xhat' y' vinf' s' ]*cB2;

% Calculate the right hand side of the constraint equations.  b1, b2, b3 all represent
% different constraint configurations.

b1 = cE*[xhat' y' vinf' u' ll' mm' ]';
b2 = cE*[xhat' y' vinf' u' ll2' mm2' ]';
b3 = cE*[xhat' y' vinf' u' ll3' mm3' ]';

if t < 0.5  % Implement constraint set 1for all times less than the time specified.
   [x,lambda,how] = qp(2*S,T1,cD,b1);
elseif t < 1.0  % Implement constraint set 2 at times less than the specified time.
   [x,lambda,how] = qp(2*S,T1,cD,b2);
else  % Implement constraint set 3 and weight set 2 at all other times
   [x,lambda,how] = qp(2*S2,T2,cD,b3);
end
```

```
% Implement the first input move.

v = x(1:xi);
```

Appendix C: Simulink Diagram



Works Cited

[1] D.W. Clarke, C. Mohtadi and P.S. Tuffs. Generalized Predictive Control, Parts 1 and 2 *Automatica*, Vol. 23, 1987, pp. 137-160.

[2] B. Kouvaritakis, J.A. Rossiter, A.O.T. Chang. Stable generalised predictive control: an algorithm with guaranteed stability, *IEE Proceedings-D*, Vol. 139, No. 4, 1992, pp. 349-362.

[3] J.A. Rossiter, B. Kouvaritakis, and J.R. Gossner. Constrained predictive control: Stability results and the case of disturbances. Submitted to *IEEE Transactions on Automatic Control*, 1994.

[4] S.A. Heise. Multivariable Constrained Model Predictive Control. PhD dissertation. Pembroke College, Cambridge, 1994.

[5] K. Zhou, J. Doyle, and K. Glover. *Robust and Optimal Control*. Upper Saddle River, NJ: Prentice Hall, 1996.

[6] J.M. Maciejowski and S.A. Heise. Heuristic Robustness Analysis of Model-Based Predictive Controllers. In *Proceedings of the 12th World Congress*. International Federation of Automatic Control, 1993.

[7] P. Voulgaris. High Performance Multivariable Control of the "Supermaneuverable" F-18/HARV Fighter Aircraft. Masters thesis. Massachusetts Insititute of Technology, 1988.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	2 DEC 1996	THESIS

4. TITLE AND SUBTITLE

MODEL PREDICTIVE CONTROL OF AEROSPACE SYSTEMS

5. FUNDING NUMBERS

6. AUTHOR(S)

CAPTAIN DEREK W. EBDON, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GAE/ENY/96D-3

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

WL/FIGC-3
Bldg. 146
2210 8th St. Ste 21
WPAFB OH 45433-7531

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

A

13. ABSTRACT *(Maximum 200 words)*

This research effort applies an MPC strategy to a high-performance aerospace system with the goal of exploiting the constraint handling qualities of MPC for fault-tolerant control. To demonstrate the effectiveness of MPC at handling control surface failures, one or more control surfaces on a high-performance fighter aircraft, primarily the F-18 High ALpha Research Vehicle, are rendered inoperable during longitudinal maneuvers. In general, simulated failures of a single control element occur at peak deflection, whereas the time and magnitude of multiple control element failures are selected on a case-by-case basis. Subsequent to the simulated failure or failures, the controller is forced to compensate by using the remaining control surfaces both to maintain stability and to attempt to retain near-nominal performance.

14. SUBJECT TERMS

Model Predictive Control, Fault-Tolerant Control, Control Systems, Aircraft Control

15. NUMBER OF PAGES

141

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	